APPENDIX A (SOURCE CODE)
to
"SYSTEM and METHODS for MANAGING DIGITAL CREATIVE WORKS"
by John S. Erickson

Filed on October 11, 1996

```
Attribute VB_Name = "MTED"
Option Explicit

Public gErr As New CErrorHandler
Public gClipboard As New Collection

Type POINTAPI
        X As Long
        y As Long
End Type

Type RECT
        Left As Long
        Top As Long
        Right As Long
        Bottom As Long
End Type

Public Const MAXSTRLEN = 255     ' max length of a prepared string

' structured stroage constants
Public Const STGM_DIRECT = &H0&
Public Const STGM_TRANSACTED = &H10000
Public Const STGM_SIMPLE = &H8000000
Public Const STGM_READ = &H0&
Public Const STGM_WRITE = &H1&
Public Const STGM_READWRITE = &H2&
Public Const STGM_SHARE_DENY_NONE = &H40&
Public Const STGM_SHARE_DENY_READ = &H30&
Public Const STGM_SHARE_DENY_WRITE = &H20&
Public Const STGM_SHARE_EXCLUSIVE = &H10&
Public Const STGM_PRIORITY = &H40000
Public Const STGM_DELETEONRELEASE = &H4000000
Public Const STGM_CREATE = &H1000&
Public Const STGM_CONVERT = &H20000
Public Const STGM_FAILIFTHERE = &H0&
Public Const STGM_NOSCRATCH = &H100000

Public Const CCHDEVICENAME = 32 ' size of a device name string
Public Const CCHFORMNAME = 32    ' size of a form name string

Type DEVMODE
        dmDeviceName As String * CCHDEVICENAME
        dmSpecVersion As Integer
        dmDriverVersion As Integer
        dmSize As Integer
        dmDriverExtra As Integer
        dmFields As Long
        dmOrientation As Integer
        dmPaperSize As Integer
        dmPaperLength As Integer
        dmPaperWidth As Integer
        dmScale As Integer
        dmCopies As Integer
        dmDefaultSource As Integer
        dmPrintQuality As Integer
        dmColor As Integer
        dmDuplex As Integer
        dmYResolution As Integer
        dmTTOption As Integer
        dmCollate As Integer
        dmFormName As String * CCHFORMNAME
```

```
        dmUnusedPadding As Integer
        dmBitsPerPel As Integer
        dmPelsWidth As Long
        dmPelsHeight As Long
        dmDisplayFlags As Long
        dmDisplayFrequency As Long
End Type

Type MEMORYSTATUS
        dwLength As Long
        dwMemoryLoad As Long
        dwTotalPhys As Long
        dwAvailPhys As Long
        dwTotalPageFile As Long
        dwAvailPageFile As Long
        dwTotalVirtual As Long
        dwAvailVirtual As Long
End Type

Public Const CWP_ALL = &H0
Public Const HWND_TOPMOST = -1
Public Const NULL_BRUSH = 5
Public Const PS_SOLID = 0
Public Const R2_XORPEN = 7       '  DPx
Public Const SWP_NOSIZE = &H1
Public Const SWP_NOMOVE = &H2
Public Const GWL_HWNDPARENT = (-8)

Declare Function BringWindowToTop Lib "user32" (ByVal hwnd As Long) As Long
Declare Function ChildWindowFromPoint Lib "user32" (ByVal hwnd As Long, ByVal
X As Long, ByVal y As Long) As Long
Declare Function ChildWindowFromPointEx Lib "user32" (ByVal hWndParent As
Long, ByVal ptx As Long, ByVal pty As Long, ByVal uFlags As Long) As Long
Declare Function ClientToScreen Lib "user32" (ByVal hwnd As Long, lpPoint As
POINTAPI) As Long
Declare Function CreateDC Lib "gdi32" Alias "CreateDCA" (ByVal lpDriverName As
String, ByVal lpDeviceName As String, ByVal lpOutput As String, lpInitData As
DEVMODE) As Long
Declare Function CreateDCByNum Lib "gdi32" Alias "CreateDCA" (ByVal
lpDriverName As String, ByVal lpDeviceName As String, ByVal lpOutput As
String, lpInitData As Long) As Long
Declare Function CreatePen Lib "gdi32" (ByVal nPenStyle As Long, ByVal nWidth
As Long, ByVal crColor As Long) As Long
Declare Function CreateSolidBrush Lib "gdi32" (ByVal crColor As Long) As Long
Declare Function DeleteDC Lib "gdi32" (ByVal hdc As Long) As Long
Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long) As Long
Declare Function DrawFocusRect Lib "user32" (ByVal hdc As Long, lpRect As
RECT) As Long
Declare Function GetClientRect Lib "user32" (ByVal hwnd As Long, lpRect As
RECT) As Long
Declare Function GetCursorPos Lib "user32" (lpPoint As POINTAPI) As Long
Declare Function GetDC Lib "user32" (ByVal hwnd As Long) As Long
Declare Function GetDesktopWindow Lib "user32" () As Long
Declare Function GetStockObject Lib "gdi32" (ByVal nIndex As Long) As Long
Declare Function GetTempFileName Lib "kernel32" Alias "GetTempFileNameA"
(ByVal lpszPath As String, ByVal lpPrefixString As String, ByVal wUnique As
Long, ByVal lpTempFileName As String) As Long
Declare Function GetTempPath Lib "kernel32" Alias "GetTempPathA" (ByVal
nBufferLength As Long, ByVal lpBuffer As String) As Long
Declare Function GetWindowRect Lib "user32" (ByVal hwnd As Long, lpRect As
RECT) As Long
Declare Sub GlobalMemoryStatus Lib "kernel32" (lpBuffer As MEMORYSTATUS)
```

```
Declare Function LockWindowUpdate Lib "user32" (ByVal hwndLock As Long) As
Long
Declare Function OffsetRect Lib "user32" (lpRect As RECT, ByVal X As Long,
ByVal y As Long) As Long
Declare Function PtInRect Lib "user32" (lpRect As RECT, ByVal X As Long, ByVal
y As Long) As Long
Declare Function Rectangle Lib "gdi32" (ByVal hdc As Long, ByVal X1 As Long,
ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long) As Long
Declare Function ReleaseDC Lib "user32" (ByVal hwnd As Long, ByVal hdc As
Long) As Long
Declare Function ScreenToClient Lib "user32" (ByVal hwnd As Long, lpPoint As
POINTAPI) As Long
Declare Function SelectObject Lib "gdi32" (ByVal hdc As Long, ByVal hObject As
Long) As Long
Declare Function SetParent Lib "user32" (ByVal hWndChild As Long, ByVal
hWndNewParent As Long) As Long
Declare Function SetPixelV Lib "gdi32" (ByVal hdc As Long, ByVal X As Long,
ByVal y As Long, ByVal crColor As Long) As Long
Declare Function SetROP2 Lib "gdi32" (ByVal hdc As Long, ByVal nDrawMode As
Long) As Long
Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, ByVal
hWndInsertAfter As Long, ByVal X As Long, ByVal y As Long, ByVal cx As Long,
ByVal cy As Long, ByVal wFlags As Long) As Long
Declare Function SetWindowWord Lib "user32" (ByVal hwnd As Long, ByVal nIndex
As Long, ByVal wNewWord As Long) As Long
Declare Function WindowFromPoint Lib "user32" (ByVal xPoint As Long, ByVal
yPoint As Long) As Long


Public Function GetTempObjectName(sPrefix As String) As String

    Dim lRetval As Long
    Dim lPos As Long
    Dim sPathRet As String
    Dim sObjNameRet As String

    ' generate a new temporary object name using temp
    ' path and temp filename API calls
    sPathRet = PrepRetStr(sPathRet)
    lRetval = GetTempPath(Len(sPathRet), sPathRet)
    sPathRet = TrimSZStr(sPathRet)
    sObjNameRet = PrepRetStr(sObjNameRet)
    lRetval = GetTempFileName(sPathRet, sPrefix, 0&, sObjNameRet)
    sObjNameRet = TrimSZStr(sObjNameRet)

    If lRetval > 0 Then
        lPos = InStr(sObjNameRet, sPathRet)
        sObjNameRet = LCase$(Mid$(sObjNameRet, lPos + Len(sPathRet),
        Len(sObjNameRet) - (lPos - 1) - Len(sPathRet)))
        sObjNameRet = Left$(sObjNameRet, InStr(sObjNameRet, ".tmp") - 1)
    End If

    GetTempObjectName = sObjNameRet

End Function


Private Function PrepRetStr(sReturn As String) As String

    ' create a string used for API calls that need a string buffer
    sReturn = String$(MAXSTRLEN, Chr$(0)) & Chr$(0)
```

```
    PrepRetStr = sReturn

End Function

Private Function TrimSZStr(sSource As String) As String

    Dim lPos As Long
    Dim sReturn As String

    ' trim a string up to a null terminator
    lPos = InStr(sSource, Chr$(0))
    If lPos > 0 Then
        sReturn = Left$(sSource, lPos - 1)
    Else
        sReturn = csEmpty
    End If

    TrimSZStr = sReturn

End Function


Private Sub DCDrawFocusRect(hwnd As Long, lpRect As RECT)

    Dim bRetval As Boolean
    Dim hdc As Long
    Dim lRetval As Long

    ' get device context for form
    hdc = GetDC(hwnd)

    bRetval = DrawFocusRect(hdc, lpRect)

    lRetval = ReleaseDC(hwnd, hdc)

End Sub
Private Sub DCDrawRect(hwnd As Long, ByVal X1 As Long, ByVal Y1 As Long, ByVal
X2 As Long, ByVal Y2 As Long)

    Dim lRetval As Long
    Dim hdc As Long
    Dim hPen As Long
    Dim hBrush As Long
    Dim hOldPen As Long
    Dim hOldBrush As Long

    ' get device context for form
    hdc = GetDC(hwnd)

    hPen = CreatePen(0, 0, QBColor(0))
    hBrush = CreateSolidBrush(QBColor(0))
    hOldPen = SelectObject(hdc, hPen)
    hOldBrush = SelectObject(hdc, hBrush)

    ' draw rectangle
    lRetval = Rectangle(hdc, X1 / Screen.TwipsPerPixelX, Y1 /
    Screen.TwipsPerPixelY, _
        X2 / Screen.TwipsPerPixelX, Y2 / Screen.TwipsPerPixelY)

    ' release device context
    If (hOldPen <> 0) And (hOldBrush <> 0) Then
```

```
        lRetval = SelectObject(hdc, hOldPen)
        lRetval = SelectObject(hdc, hOldBrush)
    End If
    lRetval = DeleteObject(hPen)
    lRetval = DeleteObject(hBrush)
    lRetval = ReleaseDC(hwnd, hdc)

End Sub


Public Function GetControlIcon(tmpControl As Control) As String

    Dim nReturn As Integer

    Select Case TypeName(tmpControl)
        Case teIDS_CtrlLabel
            nReturn = tePicControlLabel
        Case teIDS_CtrlTextBox
            Select Case Left$(tmpControl.Name, 3)
                Case "lbl"
                    nReturn = tePicControlLabel
                Case "txt"
                    nReturn = tePicControlTextbox
            End Select
        Case teIDS_CtrlCommandButton
            nReturn = tePicControlCommandButton
        Case teIDS_CtrlComboBox
            nReturn = tePicControlCombobox
        Case teIDS_CtrlListBox
            nReturn = tePicControlListbox
        Case teIDS_CtrlOptionButton
            nReturn = tePicControlOptionButton
        Case teIDS_CtrlCheckBox
            nReturn = tePicControlCheckbox
        Case teIDS_CtrlFrame
            nReturn = tePicControlFrame
        Case teIDS_CtrlPictureBox
            nReturn = tePicControlIcon
        Case Else
            nReturn = tePicControl
    End Select

    GetControlIcon = nReturn

End Function

Public Sub LoadImageList(tmpClip As PictureClip, tmpImageList As ImageList,
Optional vImageHeight As Variant, Optional vImageWidth As Variant)

    Dim i As Integer
    Dim tmpImage As ListImage

    tmpImageList.ImageHeight = IIf(IsMissing(vImageHeight), 16,
    CLng(vImageHeight))
    tmpImageList.ImageWidth = IIf(IsMissing(vImageWidth), 16,
    CLng(vImageWidth))
    For i = 0 To (tmpClip.Cols - 1)
        Set tmpImage = tmpImageList.ListImages.Add(, , tmpClip.GraphicCell(i))
    Next i

End Sub
```

```
Public Sub LockWndUpdate(hwnd As Long)

    Dim lRetval As Long

    lRetval = LockWindowUpdate(hwnd)

End Sub

Public Sub MakeWndTopmost(hwnd As Long)

    Dim lRetval As Long

    lRetval = SetWindowPos(hwnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_NOMOVE +
    SWP_NOSIZE)

End Sub

Public Sub Main()

    On Error GoTo ProcError

    Load FMain
    If FMain.LoadSuccess Then
        'Set FMain.CollectionObject = gTemplate
        'With gTemplate
        '    Set .UIForm = FMain
        '    .Name = FMain.Name & CStr(FMain.hWnd)
        '    .OID = FMain.hWnd
        'End With
        FMain.Show
    Else
        Unload FMain
        ExitApp
    End If

ProcExit:
    Exit Sub

ProcError:
    gErr.Module = "MTED"
    gErr.Proc = "Main"
    gErr.HandleError
    ExitApp

End Sub

Public Sub ExitApp()

    ' exit the application
    'Set FMain.CollectionObject = Nothing
    Set FMain.Editor = Nothing
    Set FMain = Nothing
    End

End Sub

Public Function TrimNull(sInput As String) As String

    Dim lCount As Long

    ' search input string for null terminator
    Do Until (lCount = Len(sInput))
```

```
        lCount = lCount + 1
        If (Mid$(sInput, lCount, 1) = csNullChr) Then
            Exit Do
        End If
    Loop

    TrimNull = Left$(sInput, lCount - 1)

End Function

Public Sub CenterFormToScreen(FForm As Form)

    FForm.Left = (Screen.Width - FForm.Width) / 2
    FForm.Top = (Screen.Height - FForm.Height) / 2

End Sub

Public Sub CenterFormToForm(FForm As Form, FParentForm As Form)

    If Not (FForm Is Nothing) And Not (FParentForm Is Nothing) Then
        FForm.Top = ((FParentForm.Height - FForm.Height) / 2) +
        FParentForm.Top
        FForm.Left = ((FParentForm.Width - FForm.Width) / 2) +
        FParentForm.Left
    End If

End Sub

Public Sub OffsetFormToForm(FForm As Form, FParentForm As Form, lOffset As
Long)

    If Not (FForm Is Nothing) And Not (FParentForm Is Nothing) Then
        FForm.Top = ((FParentForm.Height - FForm.Height) / 2) +
        FParentForm.Top
        FForm.Left = ((FParentForm.Width - FForm.Width) / 2) +
        FParentForm.Left
        FForm.Top = FForm.Top + lOffset
        FForm.Left = FForm.Left + lOffset
    End If

End Sub

Public Sub SelectActiveControlText()

    Dim tmpForm As Form
    Dim tmpControl As Control

    Set tmpForm = Screen.ActiveForm
    Set tmpControl = tmpForm.ActiveControl

    ' select text within a text box
    If (TypeOf tmpControl Is TextBox) Then
        tmpControl.SelLength = Len(tmpControl.Text)
    End If

End Sub

Public Sub SelectControlText(tmpControl As Control)

    ' select text within a text box
    If (TypeOf tmpControl Is TextBox) Then
        tmpControl.SelStart = 0
```

```
        tmpControl.SelLength = Len(tmpControl.Text)
    End If

End Sub
```

```
Attribute VB_Name = "MRegistry"
Option Explicit

Public Const REG_SZ As Long = 1
Public Const REG_DWORD As Long = 4
Public Const HKEY_CLASSES_ROOT = &H80000000
Public Const HKEY_CURRENT_USER = &H80000001
Public Const HKEY_LOCAL_MACHINE = &H80000002
Public Const HKEY_USERS = &H80000003
Public Const ERROR_NONE = 0
Public Const ERROR_BADDB = 1
Public Const ERROR_BADKEY = 2
Public Const ERROR_CANTOPEN = 3
Public Const ERROR_CANTREAD = 4
Public Const ERROR_CANTWRITE = 5
Public Const ERROR_OUTOFMEMORY = 6
Public Const ERROR_INVALID_PARAMETER = 7
Public Const ERROR_ACCESS_DENIED = 8
Public Const ERROR_INVALID_PARAMETERS = 87
Public Const ERROR_NO_MORE_ITEMS = 259
Public Const KEY_QUERY_VALUE = &H1
Public Const KEY_SET_VALUE = &H2
Public Const KEY_ALL_ACCESS = &H3F
Public Const REG_OPTION_NON_VOLATILE = 0

Declare Function RegCloseKey Lib "advapi32.dll" (ByVal hKey As Long) As Long
Declare Function RegCreateKeyEx Lib "advapi32.dll" Alias "RegCreateKeyExA"
(ByVal hKey As Long, ByVal lpSubKey As String, ByVal Reserved As Long, ByVal
lpClass As String, ByVal dwOptions As Long, ByVal samDesired As Long, ByVal
lpSecurityAttributes As Long, phkResult As Long, lpdwDisposition As Long) As
Long
Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias "RegOpenKeyExA" (ByVal
hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal
samDesired As Long, phkResult As Long) As Long
Declare Function RegQueryValueExString Lib "advapi32.dll" Alias
"RegQueryValueExA" (ByVal hKey As Long, ByVal lpValueName As String, ByVal
lpReserved As Long, lpType As Long, ByVal lpData As String, lpcbData As Long)
As Long
Declare Function RegQueryValueExLong Lib "advapi32.dll" Alias
"RegQueryValueExA" (ByVal hKey As Long, ByVal lpValueName As String, ByVal
lpReserved As Long, lpType As Long, lpData As Long, lpcbData As Long) As Long
Declare Function RegQueryValueExNULL Lib "advapi32.dll" Alias
"RegQueryValueExA" (ByVal hKey As Long, ByVal lpValueName As String, ByVal
lpReserved As Long, lpType As Long, ByVal lpData As Long, lpcbData As Long) As
Long
Declare Function RegSetValueExString Lib "advapi32.dll" Alias "RegSetValueExA"
(ByVal hKey As Long, ByVal lpValueName As String, ByVal Reserved As Long,
ByVal dwType As Long, ByVal lpValue As String, ByVal cbData As Long) As Long
Declare Function RegSetValueExLong Lib "advapi32.dll" Alias "RegSetValueExA"
(ByVal hKey As Long, ByVal lpValueName As String, ByVal Reserved As Long,
ByVal dwType As Long, lpValue As Long, ByVal cbData As Long) As Long
Public Sub CreateRegKey(lPredefinedKey As Long, sNewKeyName As String)

    Dim hNewKey As Long         ' handle to the new key
    Dim lRetval As Long         ' result of the RegCreateKeyEx function

    lRetval = RegCreateKeyEx(lPredefinedKey, sNewKeyName, 0&, _
              vbNullString, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, _
              0&, hNewKey, lRetval)
    RegCloseKey (hNewKey)

End Sub
```

```
Private Function SetValueEx(ByVal hKey As Long, sValueName As String, lType As
Long, vValue As Variant) As Long

    Dim lValue As Long
    Dim sValue As String

    Select Case lType
        Case REG_SZ
            sValue = vValue & Chr$(0)
            SetValueEx = RegSetValueExString(hKey, sValueName, 0&, lType,
            sValue, Len(sValue))
        Case REG_DWORD
            lValue = vValue
            SetValueEx = RegSetValueExLong(hKey, sValueName, 0&, lType,
            lValue, 4)
    End Select

End Function
Private Function QueryValueEx(ByVal hKey As Long, ByVal sValueName As String,
vValue As Variant, vReturn As Variant) As Long

    Dim lBufferLen As Long
    Dim lRetval As Long
    Dim lType As Long
    Dim lValue As Long
    Dim sValue As String

    ' determine the size and type of data to be read
    lRetval = RegQueryValueExNULL(hKey, sValueName, 0&, lType, 0&, lBufferLen)
    If (lRetval = ERROR_NONE) Then
        Select Case lType
            Case REG_SZ:              ' string
                sValue = String(lBufferLen, 0)
                lRetval = RegQueryValueExString(hKey, sValueName, 0&, _
                    lType, sValue, lBufferLen)
                vReturn = IIf(lRetval = ERROR_NONE, TrimNull(Left$(sValue,
                lBufferLen)), Empty)
            Case REG_DWORD:           ' DWORD
                lRetval = RegQueryValueExLong(hKey, sValueName, 0&, _
                    lType, lValue, lBufferLen)
                vReturn = IIf(lRetval = ERROR_NONE, lValue, 0)

            Case Else                 ' all other data types not supported
                vReturn = -1
        End Select
    End If

    QueryValueEx = lRetval

End Function
Public Sub SetRegKeyValue(lPredefinedKey As Long, sKeyName As String,
sValueName As String, vValueSetting As Variant, lValueType As Long)

    Dim lRetval As Long         ' result of the SetValueEx function
    Dim hKey As Long            ' handle of open key

    ' open the specified key
    'lRetVal = RegOpenKeyEx(HKEY_CURRENT_USER, sKeyName, 0, KEY_ALL_ACCESS,
    hKey)
    lRetval = RegOpenKeyEx(lPredefinedKey, sKeyName, 0, KEY_ALL_ACCESS, hKey)
    lRetval = SetValueEx(hKey, sValueName, lValueType, vValueSetting)
    RegCloseKey (hKey)
```

```
End Sub

Public Function GetRegKeyValue(lPredefinedKey As Long, sKeyName As String,
sValueName As String, vReturn As Variant) As Long

    Dim lRetval As Long          ' result of the API functions
    Dim hKey As Long             ' handle of opened key
    Dim vValue As Variant        ' setting of queried value

    lRetval = RegOpenKeyEx(lPredefinedKey, sKeyName, 0, KEY_ALL_ACCESS, hKey)
    If (lRetval = ERROR_NONE) Then
        lRetval = QueryValueEx(hKey, sValueName, vValue, vReturn)
    End If
    RegCloseKey (hKey)

    GetRegKeyValue = lRetval

End Function


Public Function IsRegKeyValid(lPredefinedKey As Long, sKeyName As String) As
Boolean

    Dim hKey As Long
    Dim lRetval As Long

    ' determine if a registry key is valid
    lRetval = RegOpenKeyEx(lPredefinedKey, sKeyName, 0, KEY_ALL_ACCESS, hKey)
    If (lRetval = ERROR_NONE) Then
        RegCloseKey (hKey)
    End If

    IsRegKeyValid = IIf((lRetval = ERROR_NONE), True, False)

End Function
```

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "CTemplate"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Public OID As Long
Public Name As String
Public ParentObject As CEditor
Public UIForm As New FTemplate
Public TabObject As SSIndexTab
Public TabStripObject As TabStrip
Public StorageObject As New LicensIt.Template
Public PropertySets As New Collection
Public PropertyPages As New Collection
Public StoragePath As String
Public StorageName As String
Public Dirty As Boolean

Private mnPageCount As Integer
Public Sub AddPropertyPage(Optional vPageName As Variant)

    Dim i As Integer
    Dim nIndex As Integer
    Dim sOID As String
    Dim sPageName As String
    Dim tmpPropertyPage As New CPropertyPage
    Dim tmpPage As LicensIt.LITPROPERTYPAGE
    Dim tmpPageNames As New Collection

    ' get names of current property pages
    If (Me.TabObject.Tabs.Count > 1) Then
        For i = 1 To (Me.TabObject.Tabs.Count - 1)
            tmpPageNames.Add Me.TabObject.Tabs(i).Tag,
            Me.TabObject.Tabs(i).Tag
        Next i
    End If

    ' if page name was not passed, display dialog to allow user to
    ' enter new property page name
    If (IsMissing(vPageName)) Then
        Set FNameDlg.CurrentPageNames = tmpPageNames
        FNameDlg.DlgMode = 0
        Load FNameDlg
        If (FNameDlg.LoadSuccess) Then
            FNameDlg.Show vbModal
            If (FNameDlg.PageName <> csEmpty) Then
                sPageName = FNameDlg.PageName
            End If
        End If
    Else
        sPageName = CStr(vPageName)
    End If

    If (sPageName <> csEmpty) Then
        ' if no property pages have been created in the storage object, this
        ' routine is being called when opening a template, so set index to
        zero.
        ' otherwise, get current tabcount and add a tab to the control
```

```
'         nIndex = Me.TabObject.Tabs.Count
'         Me.TabObject.Tabs.Add nIndex
'         Me.TabObject.Tabs(nIndex).Visible = True
'         'Me.TabObject.Tabs(nIndex).Tag = GetTempObjectName("ppg")
'         Me.TabObject.Tabs(nIndex).Tag = sPageName
'         Me.TabObject.Tabs(nIndex).Caption = Me.TabObject.Tabs(nIndex).Tag

        mnPageCount = mnPageCount + 1
        Load Me.UIForm.conPage(mnPageCount)
        TabStripObject.Tabs.Add mnPageCount, sPageName, sPageName

        ' add a property page to the collection
'         sOID = Me.TabObject.Tabs(nIndex).Tag
        sOID = CStr(Me.UIForm.conPage(mnPageCount).hwnd)
        Me.PropertyPages.Add tmpPropertyPage, sOID
        tmpPropertyPage.OID = sOID
        Set tmpPropertyPage.ParentObject = Me

        ' add the new property page to the storage object
        Me.StorageObject.PropertyPages.Add tmpPropertyPage.OID

        ' mark template as dirty
        Me.Dirty = True
        'UpdateTree
    End If

End Sub

Public Sub RemovePropertyPage()

    Dim i As Integer
    Dim nIndex As Integer
    Dim sOID As String
    Dim tmpPageNames As New Collection

    ' if more than one property page exists in the template, allow
    ' the user to select the one to be removed
    If (Me.PropertyPages.Count > 1) Then
        ' get names of current property pages
        If (Me.TabObject.Tabs.Count > 1) Then
            For i = 1 To (Me.TabObject.Tabs.Count - 1)
                tmpPageNames.Add Me.TabObject.Tabs(i).Tag,
                Me.TabObject.Tabs(i).Tag
            Next i
        End If

        ' display dialog so user can select property page to be removed
        Set FNameDlg.CurrentPageNames = tmpPageNames
        FNameDlg.DlgMode = 1
        Load FNameDlg
        If (FNameDlg.LoadSuccess) Then
            FNameDlg.Show vbModal
            If (FNameDlg.PageName <> csEmpty) Then
                ' determine tab index of property page selected from dialog
                For i = 1 To (Me.TabObject.Tabs.Count - 1)
                    If (Me.TabObject.Tabs(i).Tag = FNameDlg.PageName) Then
                        nIndex = i
                        Exit For
                    End If
                Next i

                ' if a valid tab index was found, confirm remove
```

```
                If (nIndex > 0) Then
                    If (MsgBox("Are you sure you want to remove this property
                    page?", _
                        vbQuestion + vbYesNo + vbDefaultButton2, "Remove
                        Property Page") = vbYes) Then
                        ' remove the property page. set current tab to index
                        and
                        ' get OID from tab's tag
                        Me.TabObject.Tab = nIndex
                        sOID = Me.TabObject.Tabs(Me.TabObject.Tab).Tag

                        ' remove the property page from the storage object
                        Me.StorageObject.PropertyPages.Remove sOID

                        ' remove the property page from the collection
                        Me.PropertyPages.Remove CStr(sOID)
                        'UpdateTree

                        ' remove tab from control
                        Me.TabObject.Tabs.Delete nIndex
                        TabStripObject.Tabs.Remove nIndex

                        ' mark template as dirty
                        Me.Dirty = True
                    End If
                End If
            End If
        End If

End Sub

Public Sub UpdateTree()

    ' update the tree form if available and visible
    'If Not (Me.UIForm.TreeForm Is Nothing) Then
    '    Me.UIForm.TreeForm.UpdateTree
    'End If

End Sub


Private Sub Class_Initialize()

    ' if template loaded successfully, then show the form.
    ' otherwise, unload it
    Load Me.UIForm
    If Me.UIForm.LoadSuccess Then
        Me.Name = Me.UIForm.Name & CStr(Me.UIForm.hwnd)
        Me.OID = Me.UIForm.hwnd
        Set Me.TabObject = Me.UIForm.sstPages
        Set Me.TabStripObject = Me.UIForm.tbsPages
        Set Me.UIForm.NTemplate = Me
        Me.UIForm.Show
        'AddPropertyPage
        Debug.Print "Template " & Me.Name & " initialized"
    Else
        Unload Me.UIForm
    End If
    mnPageCount = 1

End Sub
```

```
Private Sub Class_Terminate()

    Set Me.StorageObject = Nothing
    Debug.Print "Template terminated"

End Sub


Public Sub SaveStorage()

    ' save the template into a structured storage file
    If (Me.StorageName <> csEmpty) Then
        Me.StorageObject.Close Me.StorageName, STGM_READWRITE Or
        STGM_SHARE_EXCLUSIVE Or STGM_CREATE
        Dirty = False
    End If

End Sub


Public Sub Destroy()

    Set Me.UIForm = Nothing
    Me.ParentObject.RemoveTemplate Me

End Sub
```

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "CPropertySet"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Public OID As Long
Public Name As String
Public Properties As New Collection
```

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "CPropertyPage"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Public OID As String
Public Name As String
Public ParentObject As CTemplate
Public UIControl As Control
Public Controls As New Collection


Private Sub Class_InitializeOut()

    ' if template loaded successfully, then show the form.
    ' otherwise, unload it
'    Load Me.UIForm
'    If Me.UIForm.LoadSuccess Then
'        Me.Name = Me.UIForm.Name & CStr(Me.UIForm.hwnd)
'        Me.OID = Me.UIForm.hwnd
'        Set Me.UIForm.NPropertyPage = Me
'        Me.UIForm.Show
'        Debug.Print "PropertyPage " & Me.Name & " initialized"
'    Else
'        Unload Me.UIForm
'    End If

End Sub



Public Sub Destroy()

    'Set Me.UIForm = Nothing
    'Me.ParentObject.RemovePropertyPage Me
    'Me.ParentObject.RemovePropertyPage Me.OID

End Sub


Private Sub Class_Initialize()

End Sub

Private Sub Class_Terminate()

    Dim tmpControl As CControl

    For Each tmpControl In Me.Controls
        Set tmpControl = Nothing
    Next tmpControl
    Set Me.Controls = Nothing
    Debug.Print "PropertyPage " & CStr(Me.OID) & " terminated"

End Sub
```

```
Public Sub AddControl(tmpControl As Control, nDrawType As Integer)

    Dim ppgControl As New CControl
    Dim tmpPPGControl As LicensIt.Control

    ' add control from UI to PropertyPage's control collection
    Set ppgControl.UIControl = tmpControl
    ppgControl.TreeIcon = GetControlIcon(tmpControl)
    ppgControl.OID = tmpControl.hwnd
    ppgControl.Name = tmpControl.Name & CStr(tmpControl.hwnd)
    ppgControl.UIControl.Tag = ppgControl.Name
    Me.Controls.Add ppgControl, CStr(ppgControl.OID)

    ' add control to storage object and set properties
    Set tmpPPGControl =
    Me.ParentObject.StorageObject.PropertyPages(CStr(Me.OID)).Controls.Add(CSt
    r(ppgControl.OID))
    tmpPPGControl.Name = ppgControl.Name
    tmpPPGControl.ControlType = nDrawType
    tmpPPGControl.XPos = ppgControl.UIControl.Left
    tmpPPGControl.YPos = ppgControl.UIControl.Top
    tmpPPGControl.Height = ppgControl.UIControl.Height
    tmpPPGControl.Width = ppgControl.UIControl.Width

    Me.ParentObject.UpdateTree

End Sub

Public Sub RemoveControl(tmpControl As Control)

    ' remove control from storage object
    Me.ParentObject.StorageObject.PropertyPages(CStr(Me.OID)).Controls.Remove
    (CStr(tmpControl.hwnd))

    Me.Controls.Remove CStr(tmpControl.hwnd)
    Me.ParentObject.UpdateTree

End Sub


Public Sub UpdateControl(tmpControl As Control)

    Dim ppgControl As CControl
    Dim tmpPPGControl As LicensIt.Control

    ' get from PropertyPage's control collection
    Set ppgControl = Me.Controls(CStr(tmpControl.hwnd))

    ' get control from storage object and update properties
    Set tmpPPGControl =
    Me.ParentObject.StorageObject.PropertyPages(CStr(Me.OID)).Controls(CStr(pp
    gControl.OID))
    tmpPPGControl.XPos = ppgControl.UIControl.Left
    tmpPPGControl.YPos = ppgControl.UIControl.Top
    tmpPPGControl.Height = ppgControl.UIControl.Height
    tmpPPGControl.Width = ppgControl.UIControl.Width

    ' cleanup objects
    Set tmpPPGControl = Nothing
    Set ppgControl = Nothing
```

```
End Sub
```

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "CProperty"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Public OID As Long
Public Name As String
Public Data As Variant
Public Locator As String
```

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "CErrorHandler"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Public Module As String
Public Proc As String
Public Number As Long
Public Description As String

Private m_strProject As String


Public Sub HandleError()

    Dim bShowMsg As Boolean
    Dim sError As String

    ' set error description depending on type of error
    Number = Err.Number
    Description = Err.Description

    ' determine what to do on specific errors
    Select Case Err.Number
        Case cdlCancel          ' CANCEL button from common dialog
            ' let error handler process cancel error msg, but don't display
            bShowMsg = False
        Case Else               ' VB runtime and unanticipated errors
            bShowMsg = True
    End Select

    ' create msgbox string and display
    sError = "Error " & CStr(Number) & " in " & Err.Source & csCrLf
    sError = IIf(Module <> csEmpty, sError & Module, sError & csEmpty)
    sError = IIf(Proc <> csEmpty, sError & ":" & Proc, sError & csEmpty) &
    csCrLf
    sError = IIf(Description <> csEmpty, sError & csCrLf & Description, _
        sError & csEmpty)

    If bShowMsg Then
        MsgBox sError, vbInformation, App.Title
        Debug.Print sError
    End If

    ' Clear the project and Proc variables
    Module = csEmpty
    Proc = csEmpty
    Screen.MousePointer = vbDefault

End Sub
```

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "CEditor"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Public Templates As New Collection

Public Function AddTemplate() As CTemplate

    Dim tmpTemplate As New CTemplate

    ' add a new template to the collection
    Templates.Add tmpTemplate, CStr(tmpTemplate.OID)
    Set tmpTemplate.ParentObject = Me

    Set AddTemplate = tmpTemplate

End Function


Public Sub RemoveTemplate(tmpTemplate As CTemplate)

    ' remove template object from collection
    Templates.Remove CStr(tmpTemplate.OID)
    Set tmpTemplate = Nothing

End Sub
```

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "CControl"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Public OID As Long
Public Name As String
Public Top As Long
Public Left As Long
Public Height As Long
Public Width As Long
Public PropertyOID As Long
Public Constant As Long
Public Macro As String
Public UIControl As Control
Public TreeIcon As Integer


Private Sub Class_Terminate()

    Set UIControl = Nothing

End Sub
```

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "CClipboardObject"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Public Name As String
Public ClipObject As Object
Public Action As Integer
```

```
========================================================================
       MICROSOFT FOUNDATION CLASS LIBRARY : LitTemplate
========================================================================
```

AppWizard has created this LitTemplate DLL for you. This DLL not only demonstrates the basics of using the Microsoft Foundation classes but is also a starting point for writing your DLL.

This file contains a summary of what you will find in each of the files that make up your LitTemplate DLL.

LitTemplate.h
    This is the main header file for the DLL. It declares the CLitTemplateApp class.

LitTemplate.cpp
    This is the main DLL source file. It contains the class CLitTemplateApp. It also contains the OLE entry points required of inproc servers.

LitTemplate.odl
    This file contains the Object Description Language source code for the type library of your DLL.

LitTemplate.rc
    This is a listing of all of the Microsoft Windows resources that the program uses. It includes the icons, bitmaps, and cursors that are stored in the RES subdirectory. This file can be directly edited in Microsoft Developer Studio.

res\LitTemplate.rc2
    This file contains resources that are not edited by Microsoft Developer Studio. You should place all resources not editable by the resource editor in this file.

LitTemplate.odl
    This file contains the Object Description Language source code for the type library of your application.

LitTemplate.def
    This file contains information about the DLL that must be provided to run with Microsoft Windows. It defines parameters such as the name and description of the DLL. It also exports functions from the DLL.

LitTemplate.clw
    This file contains information used by ClassWizard to edit existing classes or add new classes. ClassWizard also uses this file to store information needed to create and edit message maps and dialog data maps and to create prototype member functions.

```
/////////////////////////////////////////////////////////////////////////////
```
Other standard files:

StdAfx.h, StdAfx.cpp
    These files are used to build a precompiled header (PCH) file named LitTemplate.pch and a precompiled types file named StdAfx.obj.

Resource.h
    This is the standard header file, which defines new resource IDs. Microsoft Developer Studio reads and updates this file.

```
////////////////////////////////////////////////////////////////////////////
Other notes:

AppWizard uses "TODO:" to indicate parts of the source code you
should add to or customize.

////////////////////////////////////////////////////////////////////////////
```

```
/* This header file machine-generated by mktyplib.exe */
/* Interface to type library: LicensIt */

#ifndef _LicensIt_H_
#define _LicensIt_H_

DEFINE_GUID(LIBID_LicensIt,0xD9592D40L,0x072C,0x11D0,0x81,0x8A,0x00,0x20,0xAF,
0xBA,0xCA,0xFF);

DEFINE_GUID(DIID_ILitTemplate,0xD9592D41L,0x072C,0x11D0,0x81,0x8A,0x00,0x20,0x
AF,0xBA,0xCA,0xFF);

/* Definition of dispatch interface: ILitTemplate */
#undef INTERFACE
#define INTERFACE ILitTemplate

DECLARE_INTERFACE_(ILitTemplate, IDispatch)
{
#ifndef NO_BASEINTERFACE_FUNCS

    /* IUnknown methods */
    STDMETHOD(QueryInterface)(THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG, AddRef)(THIS) PURE;
    STDMETHOD_(ULONG, Release)(THIS) PURE;

    /* IDispatch methods */
    STDMETHOD(GetTypeInfoCount)(THIS_ UINT FAR* pctinfo) PURE;

    STDMETHOD(GetTypeInfo)(
      THIS_
      UINT itinfo,
      LCID lcid,
      ITypeInfo FAR* FAR* pptinfo) PURE;

    STDMETHOD(GetIDsOfNames)(
      THIS_
      REFIID riid,
      OLECHAR FAR* FAR* rgszNames,
      UINT cNames,
      LCID lcid,
      DISPID FAR* rgdispid) PURE;

    STDMETHOD(Invoke)(
      THIS_
      DISPID dispidMember,
      REFIID riid,
      LCID lcid,
      WORD wFlags,
      DISPPARAMS FAR* pdispparams,
      VARIANT FAR* pvarResult,
      EXCEPINFO FAR* pexcepinfo,
      UINT FAR* puArgErr) PURE;
#endif

    /* ILitTemplate properties:
    IDispatch * PropertyPages;
    */

    /* ILitTemplate methods:
    VARIANT_BOOL Close(BSTR filename, long flags);
    VARIANT_BOOL Open(BSTR filename, long flags);
    */
```

```
};

DEFINE_GUID(IID_IDualLitTemplate,0xD9592D43L,0x072C,0x11D0,0x81,0x8A,0x00,0x20
,0xAF,0xBA,0xCA,0xFF);

/* Definition of interface: IDualLitTemplate */
#undef INTERFACE
#define INTERFACE IDualLitTemplate

DECLARE_INTERFACE_(IDualLitTemplate, IDispatch)
{
#ifndef NO_BASEINTERFACE_FUNCS

    /* IUnknown methods */
    STDMETHOD(QueryInterface)(THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG, AddRef)(THIS) PURE;
    STDMETHOD_(ULONG, Release)(THIS) PURE;

    /* IDispatch methods */
    STDMETHOD(GetTypeInfoCount)(THIS_ UINT FAR* pctinfo) PURE;

    STDMETHOD(GetTypeInfo)(
      THIS_
      UINT itinfo,
      LCID lcid,
      ITypeInfo FAR* FAR* pptinfo) PURE;

    STDMETHOD(GetIDsOfNames)(
      THIS_
      REFIID riid,
      OLECHAR FAR* FAR* rgszNames,
      UINT cNames,
      LCID lcid,
      DISPID FAR* rgdispid) PURE;

    STDMETHOD(Invoke)(
      THIS_
      DISPID dispidMember,
      REFIID riid,
      LCID lcid,
      WORD wFlags,
      DISPPARAMS FAR* pdispparams,
      VARIANT FAR* pvarResult,
      EXCEPINFO FAR* pexcepinfo,
      UINT FAR* puArgErr) PURE;
#endif

    /* IDualLitTemplate methods */
};

DEFINE_GUID(CLSID_Template,0xD9592D42L,0x072C,0x11D0,0x81,0x8A,0x00,0x20,0xAF,
0xBA,0xCA,0xFF);

#ifdef __cplusplus
class Template;
#endif

DEFINE_GUID(DIID_IControl,0xD9592D44L,0x072C,0x11D0,0x81,0x8A,0x00,0x20,0xAF,0
xBA,0xCA,0xFF);

/* Definition of dispatch interface: IControl */
#undef INTERFACE
```

```
#define INTERFACE IControl

DECLARE_INTERFACE_(IControl, IDispatch)
{
#ifndef NO_BASEINTERFACE_FUNCS

    /* IUnknown methods */
    STDMETHOD(QueryInterface)(THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG, AddRef)(THIS) PURE;
    STDMETHOD_(ULONG, Release)(THIS) PURE;

    /* IDispatch methods */
    STDMETHOD(GetTypeInfoCount)(THIS_ UINT FAR* pctinfo) PURE;

    STDMETHOD(GetTypeInfo)(
      THIS_
      UINT itinfo,
      LCID lcid,
      ITypeInfo FAR* FAR* pptinfo) PURE;

    STDMETHOD(GetIDsOfNames)(
      THIS_
      REFIID riid,
      OLECHAR FAR* FAR* rgszNames,
      UINT cNames,
      LCID lcid,
      DISPID FAR* rgdispid) PURE;

    STDMETHOD(Invoke)(
      THIS_
      DISPID dispidMember,
      REFIID riid,
      LCID lcid,
      WORD wFlags,
      DISPPARAMS FAR* pdispparams,
      VARIANT FAR* pvarResult,
      EXCEPINFO FAR* pexcepinfo,
      UINT FAR* puArgErr) PURE;
#endif

    /* IControl properties:
    short XPos;
    short YPos;
    BSTR Name;
    BSTR Label;
    short Width;
    short Height;
    BSTR OID;
    short ControlType;
    */
);

DEFINE_GUID(IID_IDualControl,0xD9592D4DL,0x072C,0x11D0,0x81,0x8A,0x00,0x20,0xA
F,0xBA,0xCA,0xFF);

/* Definition of interface: IDualControl */
#undef INTERFACE
#define INTERFACE IDualControl

DECLARE_INTERFACE_(IDualControl, IDispatch)
{
#ifndef NO_BASEINTERFACE_FUNCS
```

```
    /* IUnknown methods */
    STDMETHOD(QueryInterface)(THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG, AddRef)(THIS) PURE;
    STDMETHOD_(ULONG, Release)(THIS) PURE;

    /* IDispatch methods */
    STDMETHOD(GetTypeInfoCount)(THIS_ UINT FAR* pctinfo) PURE;

    STDMETHOD(GetTypeInfo)(
      THIS_
      UINT itinfo,
      LCID lcid,
      ITypeInfo FAR* FAR* pptinfo) PURE;

    STDMETHOD(GetIDsOfNames)(
      THIS_
      REFIID riid,
      OLECHAR FAR* FAR* rgszNames,
      UINT cNames,
      LCID lcid,
      DISPID FAR* rgdispid) PURE;

    STDMETHOD(Invoke)(
      THIS_
      DISPID dispidMember,
      REFIID riid,
      LCID lcid,
      WORD wFlags,
      DISPPARAMS FAR* pdispparams,
      VARIANT FAR* pvarResult,
      EXCEPINFO FAR* pexcepinfo,
      UINT FAR* puArgErr) PURE;
#endif

    /* IDualControl methods */
    STDMETHOD(put_XPos)(THIS_ short newXPos) PURE;
    STDMETHOD(get_XPos)(THIS_ short FAR* retval) PURE;
    STDMETHOD(put_YPos)(THIS_ short newYPos) PURE;
    STDMETHOD(get_YPos)(THIS_ short FAR* retval) PURE;
    STDMETHOD(put_Name)(THIS_ BSTR newName) PURE;
    STDMETHOD(get_Name)(THIS_ BSTR FAR* retval) PURE;
    STDMETHOD(put_Label)(THIS_ BSTR newLabel) PURE;
    STDMETHOD(get_Label)(THIS_ BSTR FAR* retval) PURE;
    STDMETHOD(put_Width)(THIS_ short newWidth) PURE;
    STDMETHOD(get_Width)(THIS_ short FAR* curWidth) PURE;
    STDMETHOD(put_Height)(THIS_ short newHeight) PURE;
    STDMETHOD(get_Height)(THIS_ short FAR* curHeight) PURE;
    STDMETHOD(put_OID)(THIS_ BSTR newOID) PURE;
    STDMETHOD(get_OID)(THIS_ BSTR FAR* retval) PURE;
    STDMETHOD(put_Type)(THIS_ short newType) PURE;
    STDMETHOD(get_Type)(THIS_ short FAR* curType) PURE;
);

DEFINE_GUID(CLSID_CONTROL,0xD9592D45L,0x072C,0x11D0,0x81,0x8A,0x00,0x20,0xAF,0
xBA,0xCA,0xFF);

#ifdef __cplusplus
class CONTROL;
#endif

DEFINE_GUID(DIID_ILitPropertyPage,0xD9592D46L,0x072C,0x11D0,0x81,0x8A,0x00,0x2
```

```
0,0xAF,0xBA,0xCA,0xFF);

/* Definition of dispatch interface: ILitPropertyPage */
#undef INTERFACE
#define INTERFACE ILitPropertyPage

DECLARE_INTERFACE_(ILitPropertyPage, IDispatch)
{
#ifndef NO_BASEINTERFACE_FUNCS

        /* IUnknown methods */
        STDMETHOD(QueryInterface)(THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
        STDMETHOD_(ULONG, AddRef)(THIS) PURE;
        STDMETHOD_(ULONG, Release)(THIS) PURE;

        /* IDispatch methods */
        STDMETHOD(GetTypeInfoCount)(THIS_ UINT FAR* pctinfo) PURE;

        STDMETHOD(GetTypeInfo)(
            THIS_
            UINT itinfo,
            LCID lcid,
            ITypeInfo FAR* FAR* pptinfo) PURE;

        STDMETHOD(GetIDsOfNames)(
            THIS_
            REFIID riid,
            OLECHAR FAR* FAR* rgszNames,
            UINT cNames,
            LCID lcid,
            DISPID FAR* rgdispid) PURE;

        STDMETHOD(Invoke)(
            THIS_
            DISPID dispidMember,
            REFIID riid,
            LCID lcid,
            WORD wFlags,
            DISPPARAMS FAR* pdispparams,
            VARIANT FAR* pvarResult,
            EXCEPINFO FAR* pexcepinfo,
            UINT FAR* puArgErr) PURE;
#endif

        /* ILitPropertyPage properties:
        BSTR Name;
        IDispatch * Controls;
        BSTR OID;
        */
};

DEFINE_GUID(IID_IDualLitPropertyPage,0xD9592D4CL,0x072C,0x11D0,0x81,0x8A,0x00,
0x20,0xAF,0xBA,0xCA,0xFF);

/* Definition of interface: IDualLitPropertyPage */
#undef INTERFACE
#define INTERFACE IDualLitPropertyPage

DECLARE_INTERFACE_(IDualLitPropertyPage, IDispatch)
{
#ifndef NO_BASEINTERFACE_FUNCS
```

```
    /* IUnknown methods */
    STDMETHOD(QueryInterface)(THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG, AddRef)(THIS) PURE;
    STDMETHOD_(ULONG, Release)(THIS) PURE;

    /* IDispatch methods */
    STDMETHOD(GetTypeInfoCount)(THIS_ UINT FAR* pctinfo) PURE;

    STDMETHOD(GetTypeInfo)(
      THIS_
      UINT itinfo,
      LCID lcid,
      ITypeInfo FAR* FAR* pptinfo) PURE;

    STDMETHOD(GetIDsOfNames)(
      THIS_
      REFIID riid,
      OLECHAR FAR* FAR* rgszNames,
      UINT cNames,
      LCID lcid,
      DISPID FAR* rgdispid) PURE;

    STDMETHOD(Invoke)(
      THIS_
      DISPID dispidMember,
      REFIID riid,
      LCID lcid,
      WORD wFlags,
      DISPPARAMS FAR* pdispparams,
      VARIANT FAR* pvarResult,
      EXCEPINFO FAR* pexcepinfo,
      UINT FAR* puArgErr) PURE;
#endif

    /* IDualLitPropertyPage methods */
    STDMETHOD(put_Name)(THIS_ VARIANT newName) PURE;
    STDMETHOD(get_Name)(THIS_ VARIANT FAR* retval) PURE;
    STDMETHOD(put_OID)(THIS_ VARIANT newName) PURE;
    STDMETHOD(get_OID)(THIS_ VARIANT FAR* retval) PURE;
    STDMETHOD(Controls)(THIS_ VARIANT FAR* retval) PURE;
);

DEFINE_GUID(CLSID_LITPROPERTYPAGE,0xD9592D47L,0x072C,0x11D0,0x81,0x8A,0x00,0x2
0,0xAF,0xBA,0xCA,0xFF);

#ifdef __cplusplus
class LITPROPERTYPAGE;
#endif

DEFINE_GUID(DIID_ILitPropertyPages,0xD9592D48L,0x072C,0x11D0,0x81,0x8A,0x00,0x
20,0xAF,0xBA,0xCA,0xFF);

/* Definition of dispatch interface: ILitPropertyPages */
#undef INTERFACE
#define INTERFACE ILitPropertyPages

DECLARE_INTERFACE_(ILitPropertyPages, IDispatch)
{
#ifndef NO_BASEINTERFACE_FUNCS

    /* IUnknown methods */
    STDMETHOD(QueryInterface)(THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
```

```
    STDMETHOD_(ULONG, AddRef)(THIS) PURE;
    STDMETHOD_(ULONG, Release)(THIS) PURE;

    /* IDispatch methods */
    STDMETHOD(GetTypeInfoCount)(THIS_ UINT FAR* pctinfo) PURE;

    STDMETHOD(GetTypeInfo)(
      THIS_
      UINT itinfo,
      LCID lcid,
      ITypeInfo FAR* FAR* pptinfo) PURE;

    STDMETHOD(GetIDsOfNames)(
      THIS_
      REFIID riid,
      OLECHAR FAR* FAR* rgszNames,
      UINT cNames,
      LCID lcid,
      DISPID FAR* rgdispid) PURE;

    STDMETHOD(Invoke)(
      THIS_
      DISPID dispidMember,
      REFIID riid,
      LCID lcid,
      WORD wFlags,
      DISPPARAMS FAR* pdispparams,
      VARIANT FAR* pvarResult,
      EXCEPINFO FAR* pexcepinfo,
      UINT FAR* puArgErr) PURE;
#endif

    /* ILitPropertyPages methods:
    IDispatch * Item(VARIANT item);
    long Count(void);
    IDispatch * Add(BSTR key);
    VARIANT Remove(VARIANT item);
    IUnknown * _NewEnum(void);
    */
);

DEFINE_GUID(CLSID_LITPROPERTYPAGES,0xD9592D49L,0x072C,0x11D0,0x81,0x8A,0x00,0x
20,0xAF,0xBA,0xCA,0xFF);

#ifdef __cplusplus
class LITPROPERTYPAGES;
#endif

DEFINE_GUID(DIID_IControls,0xD9592D4AL,0x072C,0x11D0,0x81,0x8A,0x00,0x20,0xAF,
0xBA,0xCA,0xFF);

/* Definition of dispatch interface: IControls */
#undef INTERFACE
#define INTERFACE IControls

DECLARE_INTERFACE_(IControls, IDispatch)
{
#ifndef NO_BASEINTERFACE_FUNCS

    /* IUnknown methods */
    STDMETHOD(QueryInterface)(THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG, AddRef)(THIS) PURE;
```

```
    STDMETHOD_(ULONG, Release)(THIS) PURE;

    /* IDispatch methods */
    STDMETHOD(GetTypeInfoCount)(THIS_ UINT FAR* pctinfo) PURE;

    STDMETHOD(GetTypeInfo)(
      THIS_
      UINT itinfo,
      LCID lcid,
      ITypeInfo FAR* FAR* pptinfo) PURE;

    STDMETHOD(GetIDsOfNames)(
      THIS_
      REFIID riid,
      OLECHAR FAR* FAR* rgszNames,
      UINT cNames,
      LCID lcid,
      DISPID FAR* rgdispid) PURE;

    STDMETHOD(Invoke)(
      THIS_
      DISPID dispidMember,
      REFIID riid,
      LCID lcid,
      WORD wFlags,
      DISPPARAMS FAR* pdispparams,
      VARIANT FAR* pvarResult,
      EXCEPINFO FAR* pexcepinfo,
      UINT FAR* puArgErr) PURE;
#endif

    /* IControls methods:
    IDispatch * Item(VARIANT item);
    long Count(void);
    IDispatch * Add(BSTR key);
    VARIANT Remove(VARIANT item);
    IUnknown * _NewEnum(void);
    */
);

DEFINE_GUID(CLSID_CONTROLS,0xD9592D4BL,0x072C,0x11D0,0x81,0x8A,0x00,0x20,0xAF,
0xBA,0xCA,0xFF);

#ifdef __cplusplus
class CONTROLS;
#endif

#endif
```

```
/////////////////////////////////////////////////////////////////////////////
//
// Template.h:      header file
// Abstract:        This class implements the Template object - the primary
//                  automation server for the DLL.
//
//  Date      By    Comments
// -------  ------  --------
// 13sep96    npt   Initial Version
//
/////////////////////////////////////////////////////////////////////////////

#ifndef __TEMPLATE_H__
#define __TEMPLATE_H__


/////////////////////////////////////////////////////////////////////////////
// CLitTemplate command target

class CLitTemplate : public CCmdTarget
{
    DECLARE_DYNCREATE(CLitTemplate)

    CLitTemplate();           // protected constructor used by dynamic
    creation

// Attributes
public:

private:
    CLitPropertyPages   m_PPages ;  // collection of property pages.
    DWORD               m_Version ; // Template version


// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CLitTemplate)
    public:
    virtual void OnFinalRelease();
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CLitTemplate();

protected:

    // Generated message map functions
    //{{AFX_MSG(CLitTemplate)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG

    DECLARE_MESSAGE_MAP()
    DECLARE_OLECREATE(CLitTemplate)

    // Generated OLE dispatch map functions
    //{{AFX_DISPATCH(CLitTemplate)
    LPDISPATCH m_propertyPages;
    afx_msg void OnPropertyPagesChanged();
```

```
    afx_msg BOOL Close(LPCTSTR filename, long flags);
    afx_msg BOOL Open(LPCTSTR filename, long flags);
    //}}AFX_DISPATCH
    DECLARE_DISPATCH_MAP()
    DECLARE_INTERFACE_MAP()

// 05sep95  npt  Start Dual interface support
    BEGIN_DUAL_INTERFACE_PART(DualLitTemplate, IDualLitTemplate)
    END_DUAL_INTERFACE_PART(DualLitTemplate)
// End Dual interface support

    //      add declaration of ISupportErrorInfo implementation
    //      to indicate we support the OLE Automation error object
    DECLARE_DUAL_ERRORINFO()
};

/////////////////////////////////////////////////////////////////////////////
#endif  //__TEMPLATE_H__
```

```
////////////////////////////////////////////////////////////////////////////
//
// Template.cpp:    implementation file
// Abstract:        This class implements the Template object - the primary
//                  automation server for the DLL.
//
//  Date      By    Comments
// -------  ------  --------
// 13sep96    npt   Initial Version
//
////////////////////////////////////////////////////////////////////////////

#include    "stdafx.h"
#include    "LitTemplate.h"
#include    "Control.h"
#include    "Controls.h"
#include    "LitPropPage.h"
#include    "PropPages.h"
#include    "Template.h"


#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////////////////////////////////////////////
// CLitTemplate

IMPLEMENT_DYNCREATE(CLitTemplate, CCmdTarget)

CLitTemplate::CLitTemplate()
{
    EnableAutomation();

    m_Version = VERSION1 ;
// Record IDispatch for CLitPropertyPages (property pages collection)
    m_propertyPages = m_PPages.GetIDispatch(FALSE) ;


    // To keep the application running as long as an OLE automation
    //  object is active, the constructor calls AfxOleLockApp.
    AfxOleLockApp();
}

CLitTemplate::~CLitTemplate()
{
    // To terminate the application when all objects created with
    //  with OLE automation, the destructor calls AfxOleUnlockApp.

    AfxOleUnlockApp();
}


void CLitTemplate::OnFinalRelease()
{
    // When the last reference for an automation object is released
    // OnFinalRelease is called.  The base class will automatically
    // deletes the object.  Add additional cleanup required for your
    // object before calling the base class.
```

```
    CCmdTarget::OnFinalRelease();
}


BEGIN_MESSAGE_MAP(CLitTemplate, CCmdTarget)
    //{{AFX_MSG_MAP(CLitTemplate)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_DISPATCH_MAP(CLitTemplate, CCmdTarget)
    //{{AFX_DISPATCH_MAP(CLitTemplate)
    DISP_PROPERTY_NOTIFY(CLitTemplate, "PropertyPages", m_propertyPages,
    OnPropertyPagesChanged, VT_DISPATCH)
    DISP_FUNCTION(CLitTemplate, "Close", Close, VT_BOOL, VTS_BSTR VTS_I4)
    DISP_FUNCTION(CLitTemplate, "Open", Open, VT_BOOL, VTS_BSTR VTS_I4)
    //}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()

// Note: we add support for IID_ILitTemplate to support typesafe binding
//  from VBA.  This IID must match the GUID that is attached to the
//  dispinterface in the .ODL file.

// {D9592D41-072C-11D0-818A-0020AFBACAFF}
//static const IID IID_ILitTemplate =
//{ 0xd9592d41, 0x072c, 0x11d0, {0x81, 0x8a, 0x00, 0x20, 0xaf, 0xba, 0xca,
0xff} };

BEGIN_INTERFACE_MAP(CLitTemplate, CCmdTarget)
//  INTERFACE_PART(CLitTemplate, IID_ILitTemplate, Dispatch)
    INTERFACE_PART(CLitTemplate, DIID_ILitTemplate, Dispatch)
// 05sep95  npt  Start Dual interface support
    INTERFACE_PART(CLitTemplate, IID_IDualLitTemplate, DualLitTemplate)
    DUAL_ERRORINFO_PART(CLitTemplate)
// End Dual interface support
END_INTERFACE_MAP()

// This GUID must match the GUID in the .ODL file for the "coclass"
// {D9592D42-072C-11D0-818A-0020AFBACAFF}
IMPLEMENT_OLECREATE(CLitTemplate, "LicensIt.Template", 0xd9592d42, 0x072c,
0x11d0, 0x81, 0x8a, 0x00, 0x20, 0xaf, 0xba, 0xca, 0xff)

/////////////////////////////////////////////////////////////////////////////
// CLitTemplate message handlers

/////////////////////////////////////////////////////////////////////////////
// DUAL INTERFACE FUNCTIONS
/////////////////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////////////
// delegate standard IDispatch methods to MFC IDispatch implementation
// Macro defined in MFCDUAL.H
DELEGATE_DUAL_INTERFACE(CLitTemplate, DualLitTemplate)


// Implement ISupportErrorInfo to indicate we support the
// OLE Automation error handler.
IMPLEMENT_DUAL_ERRORINFO(CLitTemplate, IID_IDualLitTemplate)



void CLitTemplate::OnPropertyPagesChanged()
```

```
{
    // TODO: Add notification handler code

}

//*****************************************************************************
// Close -  Create a structured storage file to store all objects into.
//*****************************************************************************
BOOL CLitTemplate::Close(LPCTSTR filename, long flags)
{
    USES_CONVERSION;
    BOOL        retval = FALSE ;
    LPSTORAGE   pStgRoot = NULL ;
    LPSTORAGE   pStgPropPages = NULL ;
//  const   char*   szPropPages = "LitPropertyPages" ;
//  const   char*   szTemplVer  = "LicensIt Template Version" ;
    char        buf[80] ;

    if (::StgCreateDocfile(T2OLE(filename), flags, 0, &pStgRoot) == S_OK)
        {
// Write the template version number to the structured storage
        LPSTREAM        pStream = NULL ;

        LoadString(AfxGetInstanceHandle(), IDS_STREAM_TEMPLATE_VERSION,
        (LPTSTR)&buf, 80) ;
// Create a stream to write the version number to.
//      VERIFY(pStgRoot->CreateStream(T2OLE(szTemplVer),
        VERIFY(pStgRoot->CreateStream(T2OLE(buf),
                STGM_CREATE | STGM_READWRITE | STGM_SHARE_EXCLUSIVE,
                0, 0, &pStream ) == S_OK);
        ASSERT(pStream != NULL) ;
        pStream->Write((DWORD *)&m_Version,   sizeof(DWORD), NULL) ;
        pStream->Release() ;

// The Storage file has been created - create a storage for
// property pages and write them.
        LoadString(AfxGetInstanceHandle(), IDS_STORAGE_PROPERTY_PAGES,
        (LPTSTR)&buf, 80) ;
//      VERIFY(pStgRoot->CreateStorage(T2OLE(szPropPages),
        VERIFY(pStgRoot->CreateStorage(T2OLE(buf),
            STGM_CREATE | STGM_READWRITE | STGM_SHARE_EXCLUSIVE,
            0, 0, &pStgPropPages) == S_OK) ;

// Store all the property pages
        m_PPages.Close(pStgPropPages) ;

        pStgPropPages->Release() ;
        pStgRoot->Release() ;
        retval = TRUE ;
        }

    return retval ;
}


//*****************************************************************************
// Open -   Open a structured storage file and read its contents.
//*****************************************************************************
BOOL CLitTemplate::Open(LPCTSTR filename, long flags)
{

    USES_CONVERSION;
```

```
    LPSTORAGE   pStgRoot = NULL ;
    LPSTORAGE   pStgPropPages = NULL ;
    LPSTREAM    pStream = NULL ;
    char        buf[80] ;
    DWORD       version ;

    if (::StgOpenStorage(T2OLE(filename), NULL, flags, NULL, 0L, &pStgRoot) ==
    S_OK)
        {
        // Open the "LicensIt Template Version" stream and read the version.
        LoadString(AfxGetInstanceHandle(), IDS_STREAM_TEMPLATE_VERSION,
        (LPTSTR)&buf, 80) ;
        VERIFY(pStgRoot->OpenStream(T2OLE(buf), NULL,
                        STGM_READ | STGM_SHARE_EXCLUSIVE, 0L, &pStream ) ==
                        S_OK) ;
        ASSERT(pStream != NULL) ;
        m_Version = 0L ;
        pStream->Read((DWORD *)&version,   sizeof(DWORD), NULL) ;
        pStream->Release() ;

        // Insure we have a version we can understand
        if (version <= VERSION1)
            {
            m_Version = version ;
            // Read the Property Pages
            LoadString(AfxGetInstanceHandle(), IDS_STORAGE_PROPERTY_PAGES,
            (LPTSTR)&buf, 80) ;
            VERIFY(pStgRoot->OpenStorage(T2OLE(buf), NULL,
                    STGM_READ | STGM_SHARE_EXCLUSIVE, NULL, 0L, &pStgPropPages
                    ) == S_OK) ;
            ASSERT(pStgPropPages != NULL) ;

            // Read each of the property pages
            m_PPages.Open(pStgPropPages) ;

            pStgPropPages->Release() ;
            }

        pStgRoot->Release() ;
        }


    return TRUE;
}
```

```
////////////////////////////////////////////////////////////////////////////
//
// stdafx.h:    include file for standard system include files,
//              or project specific include files that are used frequently,
but
//              are changed infrequently
//
//  Date      By    Comments
// -------  ------  --------
// 13sep96    npt   Initial Version
//
////////////////////////////////////////////////////////////////////////////


#define     VC_EXTRALEAN        // Exclude rarely-used stuff from Windows
headers

#include    <afxwin.h>         // MFC core and standard components
#include    <afxext.h>         // MFC extensions

#ifndef _AFX_NO_OLE_SUPPORT
#include    <afxole.h>         // MFC OLE classes
#include    <afxodlgs.h>       // MFC OLE dialog classes
#include    <afxdisp.h>        // MFC OLE automation classes
#endif // _AFX_NO_OLE_SUPPORT


#ifndef _AFX_NO_DB_SUPPORT
#include    <afxdb.h>           // MFC ODBC database classes
#endif // _AFX_NO_DB_SUPPORT

#ifndef _AFX_NO_DAO_SUPPORT
#include    <afxdao.h>          // MFC DAO database classes
#endif // _AFX_NO_DAO_SUPPORT

#ifndef _AFX_NO_AFXCMN_SUPPORT
#include    <afxcmn.h>          // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT


// 06sep95  npt  Start Dual interface support
#include    "TPLDual.h"     // UUID definitions created by MKTYPLIB

#include    "initiids.h"    // Global helper functions.

//   pick up the definition of AfxOleRegisterTypeLib
#include    <afxctl.h>

//   include our macros to simplify dual interface support
#include    "mfcdual.h"
// End Dual interface support

#include    <afxtempl.h>        // Required for CMap template class
#include    <afxpriv.h>         // for OLE2CT etc.

////////////////////////////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////////////////////////////
//
//  stdafx.cpp:     source file that includes just the standard includes
//  LitTemplate.pch will be the pre-compiled header
//  stdafx.obj will contain the pre-compiled type information
//
//  Date      By    Comments
// -------  ------  --------
// 13sep96    npt   Initial Version
//
////////////////////////////////////////////////////////////////////////////

#include    "stdafx.h"
```

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by LitTemplate.rc
//
#define IDS_STORAGE_PROPERTY_PAGES      1
#define IDS_STREAM_TEMPLATE_VERSION     2
#define IDS_NULL_STRING                 3
#define IDS_STREAM_PAGE_NAME            4
#define IDS_STORAGE_CONTROLS            5

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        136
#define _APS_NEXT_COMMAND_VALUE         32771
#define _APS_NEXT_CONTROL_VALUE         1000
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

```
////////////////////////////////////////////////////////////////////////////
//
// FileName:     PropPages.h
// Abstract:     This class implements the Property Pages collection object.
//
//  Date       By     Comments
// -------  ------  --------
// 13sep96    npt    Initial Version
//
////////////////////////////////////////////////////////////////////////////

// Map (Dictionary collection) to hold collection of property pages.
typedef CMap<CString, LPCSTR, CLitPropertyPage*, CLitPropertyPage*>
CPropertyPageMap ;
typedef CList<CLitPropertyPage*, CLitPropertyPage*> CPropertyPageList ;

////////////////////////////////////////////////////////////////////////////
// CLitPropertyPages command target
class CLitPropertyPages : public CCmdTarget
{
    DECLARE_DYNCREATE(CLitPropertyPages)

public:
    CLitPropertyPages();           // protected constructor used by dynamic
    creation

// Attributes
public:
    CPropertyPageList   m_PropertyPages ;   // collection of Property Pages
    POSITION            m_CurrentPosition ;

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CLitPropertyPages)
    public:
    virtual void OnFinalRelease();
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CLitPropertyPages();

    // Generated message map functions
    //{{AFX_MSG(CLitPropertyPages)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG

    DECLARE_MESSAGE_MAP()
    // Generated OLE dispatch map functions
    //{{AFX_DISPATCH(CLitPropertyPages)
    afx_msg LPDISPATCH GetItem(const VARIANT FAR& item);
    afx_msg long GetCount();
    afx_msg LPDISPATCH Add(LPCTSTR key);
    afx_msg VARIANT Remove(const VARIANT FAR& item);
    //}}AFX_DISPATCH
    DECLARE_DISPATCH_MAP()
    afx_msg LPUNKNOWN _NewEnum(void);
```

```
// Create an enumerator class to support the automation collection
    BEGIN_INTERFACE_PART(EnumVARIANT, IEnumVARIANT)
        STDMETHOD(Next)(THIS_ unsigned long celt, VARIANT FAR* rgvar, unsigned
        long FAR* pceltFetched) ;
        STDMETHOD(Skip)(THIS_ unsigned long celt) ;
        STDMETHOD(Reset)(THIS) ;
        STDMETHOD(Clone)(THIS_ IEnumVARIANT FAR* FAR* ppenum) ;
        XEnumVARIANT() ;
    END_INTERFACE_PART(EnumVARIANT)

    DECLARE_INTERFACE_MAP()

public:
    BOOL        Close(LPSTORAGE pStg) ; // Write the collection to OLE storage
    BOOL        Open (LPSTORAGE pStg) ; // Read the collection from OLE
    storage
    POSITION    Lookup(LPCSTR key) ;    // find a Property Page with the given
    key
    void        AddPagetoList(CLitPropertyPage* newPropPage) ;

};

////////////////////////////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////////////////////////////
//
// FileName:     PropPages.cpp
// Abstract:     This class implements the Property Pages collection object.
//
//  Date       By     Comments
// -------  ------  --------
// 13sep96    npt    Initial Version
//
////////////////////////////////////////////////////////////////////////////

#include    "stdafx.h"
#include    "littemplate.h"
#include    "Control.h"
#include    "Controls.h"
#include    "LitPropPage.h"
#include    "PropPages.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


////////////////////////////////////////////////////////////////////////////
// CLitPropertyPages

IMPLEMENT_DYNCREATE(CLitPropertyPages, CCmdTarget)

CLitPropertyPages::CLitPropertyPages()
{
    EnableAutomation();
}

CLitPropertyPages::~CLitPropertyPages()
{

// Remove and delete all controls in the collection
    long    Count = m_PropertyPages.GetCount() ;
    if (Count > 0)
        {
        POSITION pos = m_PropertyPages.GetHeadPosition();
        while( pos != NULL )
            {
            CLitPropertyPage*   pPropPage ;
            pPropPage = m_PropertyPages.GetNext(pos) ;
            while (pPropPage->ExternalRelease())
                ;
            }
        }

// If we have any pages in our collection, remove 'em
    m_PropertyPages.RemoveAll() ;

}


void CLitPropertyPages::OnFinalRelease()
{
    // When the last reference for an automation object is released
    // OnFinalRelease is called.  The base class will automatically
```

```
    // deletes the object.  Add additional cleanup required for your
    // object before calling the base class.

    CCmdTarget::OnFinalRelease();
}


BEGIN_MESSAGE_MAP(CLitPropertyPages, CCmdTarget)
    //{{AFX_MSG_MAP(CLitPropertyPages)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_DISPATCH_MAP(CLitPropertyPages, CCmdTarget)
    //{{AFX_DISPATCH_MAP(CLitPropertyPages)
    DISP_FUNCTION(CLitPropertyPages, "Item", GetItem, VT_DISPATCH,
    VTS_VARIANT)
    DISP_DEFVALUE(CLitPropertyPages, "Item")
    DISP_FUNCTION(CLitPropertyPages, "Count", GetCount, VT_I4, VTS_NONE)
    DISP_FUNCTION(CLitPropertyPages, "Add", Add, VT_DISPATCH, VTS_BSTR)
    DISP_FUNCTION(CLitPropertyPages, "Remove", Remove, VT_VARIANT,
    VTS_VARIANT)
    //}}AFX_DISPATCH_MAP
    DISP_PROPERTY_EX_ID(CLitPropertyPages, "_NewEnum", DISPID_NEWENUM,
    _NewEnum, SetNotSupported, VT_UNKNOWN)
END_DISPATCH_MAP()

// Note: we add support for IID_ILitPropertyPages to support typesafe binding
//  from VBA.  This IID must match the GUID that is attached to the
//  dispinterface in the .ODL file.
BEGIN_INTERFACE_MAP(CLitPropertyPages, CCmdTarget)
    INTERFACE_PART(CLitPropertyPages, DIID_ILitPropertyPages, Dispatch)
    INTERFACE_PART(CLitPropertyPages, IID_IEnumVARIANT, EnumVARIANT)
END_INTERFACE_MAP()


//////////////////////////////////////////////////////////////////////////
// Close -  For each Property Page, create a sub storage in the OLE
//          Storage in which to store the page.
//////////////////////////////////////////////////////////////////////////
BOOL CLitPropertyPages::Close(LPSTORAGE pStg)
{

// Store each property page in it's own Ole Storage
    USES_CONVERSION;
    LPSTORAGE   pStgPropPage = NULL ;
    POSITION    pos = m_PropertyPages.GetHeadPosition();
    while( pos != NULL )
        {
        CLitPropertyPage*   pPropPage ;
        LPTSTR              ptr ;

        pPropPage = m_PropertyPages.GetNext(pos);
        pPropPage->AssertValid() ;

        ptr = pPropPage->key.LockBuffer();
        VERIFY(pStg->CreateStorage(T2OLE(ptr),
            STGM_CREATE | STGM_READWRITE | STGM_SHARE_EXCLUSIVE,
            0, 0, &pStgPropPage) == S_OK) ;
        pPropPage->key.UnlockBuffer() ;
        ASSERT(pStgPropPage != NULL) ;
        pPropPage->Close(pStgPropPage) ;
```

```
        pStgPropPage->Release() ;
        }

    return TRUE ;

}

//*************************************************************************
// Open -   Read each Property Page from a sub storage in the OLE
//          Storage file.
//*************************************************************************
BOOL CLitPropertyPages::Open(LPSTORAGE pStg)
{

// Read each property page from Ole Storage
    USES_CONVERSION;
    LPSTORAGE           pStgPropPage = NULL ;
    LPENUMSTATSTG       pEnum = NULL ;
    LPMALLOC            pMalloc = NULL ;
    STATSTG             statstg ;
    CLitPropertyPage*   newPropPage ;

    ::CoGetMalloc(MEMCTX_TASK, &pMalloc) ;  // Assumes AfxOleInit was called
    VERIFY(pStg->EnumElements(0, NULL, 0, &pEnum) == S_OK) ;
    while (pEnum->Next(1, &statstg, NULL) == S_OK)
        {
        if (statstg.type == STGTY_STORAGE)
            {
            VERIFY(pStg->OpenStorage(statstg.pwcsName, NULL,
                                     STGM_READ | STGM_SHARE_EXCLUSIVE,
                                     NULL, 0, &pStgPropPage) == S_OK) ;
            ASSERT(pStgPropPage != NULL) ;
            newPropPage = new CLitPropertyPage ;

            newPropPage->Open(pStgPropPage) ;

            AddPagetoList(newPropPage) ;        // Add the page to the list
            pStgPropPage->Release() ;
            pMalloc->Free(statstg.pwcsName) ;   // avoids memory leaks
            }
        }
    pMalloc->Release() ;
    pEnum->Release() ;
    return TRUE ;
}


////////////////////////////////////////////////////////////////////////////
// CLitPropertyPages message handlers


//*************************************************************************
// _NewEnum -   Locate and return the IUnknown for the enumerator class.
//*************************************************************************
LPUNKNOWN CLitPropertyPages::_NewEnum(void)
{

//    LPUNKNOWN punkEnumVariant = (LPUNKNOWN)&m_xEnumVARIANT ;

    LPUNKNOWN   punkEnumVariant ;
    m_xEnumVARIANT.QueryInterface(IID_IUnknown, (LPVOID *)&punkEnumVariant) ;
```

```
    m_CurrentPosition = m_PropertyPages.GetHeadPosition() ;

    return punkEnumVariant;

}


//*****************************************************************************
// GetCount -   Return the number of pages in the collection.
//*****************************************************************************
long CLitPropertyPages::GetCount()
{

    return (long)m_PropertyPages.GetCount() ;

}

//*****************************************************************************
// GetItem -    Locate a CLitPropertyPage object in our collection and return
//              an IDispatch Interface pointer if found.
//  Args:       The input argument is a VARIANT specifying either the key
//              name of the desired page or an index number of a page.
//  Return:     NULL if entry is not in the collection (either the name was
//              not found or the index is out of bounds).
//              IDispatch pointer when the item is found.
//*****************************************************************************
LPDISPATCH CLitPropertyPages::GetItem(const VARIANT FAR& item)
{

    LPDISPATCH          lpDisp = NULL ;
    CString             tmpCString ;
    CLitPropertyPage*   tmpPropPage = NULL ;
    POSITION            pos ;

    if (item.vt == VT_BSTR)
        {

        CString             key = item.bstrVal ;
        pos = Lookup((LPCSTR)key) ;
        if (pos != NULL)
            {
            tmpPropPage = m_PropertyPages.GetAt(pos) ;
            ASSERT(tmpPropPage != NULL) ;
            lpDisp = tmpPropPage->GetIDispatch(TRUE) ;        // AddRef
            }
        }
    else
        {
        // coerce to VT_I4
        VARIANT     va ;
        VariantInit( &va ) ;
        if (SUCCEEDED(VariantChangeType( &va, (VARIANT FAR*)&item, 0, VT_I4
        )))
            {
            long    Counter = va.lVal ;
            if (Counter <= (long)m_PropertyPages.GetCount())
                {
                pos = m_PropertyPages.GetHeadPosition();
                while(( pos != NULL ) && (Counter > 0))
                    {
                    tmpPropPage = m_PropertyPages.GetNext(pos) ;
                    Counter-- ;
```

```
                    if (Counter == 0)
                        {
                        lpDisp = tmpPropPage->GetIDispatch(TRUE) ;        //
                        AddRef
                        }
                    }
                }
            }

    // BUGBUG:  Implement dispatch exception if lpDisp == NULL
    return lpDisp ;

}


//*****************************************************************************
// Lookup - Lookup the PropertyPage with the key in the m_PropertyPages
//          collection.
//  Return: POSITION in the CList if found, NULL otherwise.
//*****************************************************************************
POSITION CLitPropertyPages::Lookup(LPCSTR key)
{
    POSITION    retPos = NULL ;
    POSITION    pos    = m_PropertyPages.GetHeadPosition();
    POSITION    tmp ;

    while( pos != NULL )
        {
        tmp = pos ;
        CLitPropertyPage*   pPropPage = m_PropertyPages.GetNext(pos) ;
        pPropPage->AssertValid() ;
        if (lstrcmp(key, pPropPage->key) == 0)
            {
            retPos = tmp ;
            pos = NULL ;
            }
        }

    return retPos ;
}


//*****************************************************************************
// Add -    Add a new PropertyPage to the dictionary using "key" as the
//          index.
//  Return: NULL if an entry already exists with the specified key.
//          IDispatch pointer on successful addition.
//*****************************************************************************
LPDISPATCH CLitPropertyPages::Add(LPCTSTR key)
{
    LPDISPATCH          lpDisp = NULL ;
    CString             tmp = key ;
    POSITION            pos = Lookup((LPCSTR)tmp) ;

    if ( pos == NULL )
        {
        CString             pageName = key ;
        CLitPropertyPage*   newPropPage = new CLitPropertyPage(pageName) ;
        newPropPage->key = tmp ;
        m_PropertyPages.AddTail(newPropPage) ;
        lpDisp = newPropPage->GetIDispatch(TRUE) ;
```

```
        }
    return lpDisp ;
}

//*****************************************************************************
// AddPagetoList -  Add the passed in PropertyPage to the collection
//*****************************************************************************
void    CLitPropertyPages::AddPagetoList(CLitPropertyPage* newPropPage)
{
    CString             tmp = newPropPage->key ;
    POSITION            pos = Lookup((LPCSTR)tmp) ;
    LPDISPATCH          lpDisp = NULL ;

    if ( pos == NULL )
        {
        m_PropertyPages.AddTail(newPropPage) ;
        lpDisp = newPropPage->GetIDispatch(TRUE) ;  // AddRef
        }
    return ;
}


//*****************************************************************************
// Remove - Remove the PropertyPage indicated by the input VARIANT.
//          (Remove using either the key name or an index).
//  Return: VT_EMPTY VARIANT.
//*****************************************************************************
VARIANT CLitPropertyPages::Remove(const VARIANT FAR& item)
{

    CString             tmpCString ;
    CLitPropertyPage*   tmpPropPage = NULL ;
    POSITION            pos ;

    if (item.vt == VT_BSTR)
        {

        CString     key = item.bstrVal ;
        pos = Lookup((LPCSTR)key) ;
        if (pos != NULL)
            {
            tmpPropPage = m_PropertyPages.GetAt(pos) ;
            m_PropertyPages.RemoveAt(pos) ;
            // Remove all references to the object so it gets destroyed
            while (tmpPropPage->ExternalRelease())
                ;
            }
        }
    else
        {
        // coerce to VT_I4
        VARIANT     va ;
        VariantInit( &va ) ;
        if (SUCCEEDED(VariantChangeType( &va, (VARIANT FAR*)&item, 0, VT_I4
        )))
            {
            long    Counter = va.lVal ;
            if (Counter <= (long)m_PropertyPages.GetCount())
                {
                pos = m_PropertyPages.GetHeadPosition() ;
                while(( pos != NULL ) && (Counter > 0))
                    {
```

```
                    POSITION    tmp = pos ;
                    tmpPropPage = m_PropertyPages.GetNext(pos) ;
                    Counter-- ;
                    if (Counter == 0)
                        {
                        m_PropertyPages.RemoveAt(tmp) ;
                        // Remove all references to the object so it gets
                        destroyed
                        while (tmpPropPage->ExternalRelease())
                            ;
                        }

                    }
                }
            }
        }

    VARIANT     vaResult ;
    VariantInit(&vaResult) ;
    vaResult.vt = VT_EMPTY ;
    return vaResult ;
}

//*************************************************************************
//
// enumerator class XEnumVARIANT
//
//*************************************************************************

CLitPropertyPages::XEnumVARIANT::XEnumVARIANT()
{
    METHOD_PROLOGUE(CLitPropertyPages, EnumVARIANT)
}

ULONG FAR EXPORT CLitPropertyPages::XEnumVARIANT::AddRef()
{
    METHOD_PROLOGUE(CLitPropertyPages, EnumVARIANT)
    return pThis->ExternalAddRef();
}

ULONG FAR EXPORT CLitPropertyPages::XEnumVARIANT::Release()
{
    METHOD_PROLOGUE(CLitPropertyPages, EnumVARIANT)
    return pThis->ExternalRelease();
}

HRESULT FAR EXPORT CLitPropertyPages::XEnumVARIANT::QueryInterface(REFIID iid,
void FAR* FAR* ppvObj)
{
    METHOD_PROLOGUE(CLitPropertyPages, EnumVARIANT)
    return (HRESULT)pThis->ExternalQueryInterface(&iid, ppvObj);
}


//*************************************************************************
// XEnumVARIANT::Next - Retrieve the next element in the collection
//                      (position maintained in pThis->m_CurrentPosition).
//                      and return an IDispatch.
//*************************************************************************
STDMETHODIMP CLitPropertyPages::XEnumVARIANT::Next(ULONG celt,
                        VARIANT FAR* rgvar, ULONG FAR* pceltFetched)
{
```

```
    // This sets up the "pThis" pointer so that it points to our
    // containing CLitPropertyPages instance
    METHOD_PROLOGUE(CLitPropertyPages, EnumVARIANT)

    HRESULT             hr = ResultFromScode( S_FALSE ) ;
    ULONG               l  ;
    CString             tmpCString ;
    CLitPropertyPage*   tmpPropPage = NULL ;


// pceltFetched can legally == 0
    if (pceltFetched != NULL)
        {
        *pceltFetched = 0 ;
        }
    else if (celt > 1)
        {
        return ResultFromScode( E_INVALIDARG ) ;
        }

    if (pThis->m_CurrentPosition == NULL)
        {
        return ResultFromScode( S_FALSE ) ;
        }

    for (l = 0; l < celt; l++)
        {
        VariantInit( &rgvar[l] ) ;
        }


    // Retrieve the next celt elements.
    hr = NOERROR ;
    for (l = 0 ; pThis->m_CurrentPosition != NULL && celt != 0 ; l++)
        {
        tmpPropPage = pThis->m_PropertyPages.GetNext(pThis->m_CurrentPosition)
        ;
        celt-- ;
        rgvar[l].vt = VT_DISPATCH ;
        rgvar[l].pdispVal = tmpPropPage->GetIDispatch(TRUE) ;
        if (pceltFetched != NULL)
            (*pceltFetched)++ ;
        }

    return hr ;
}

//*********************************************************************
// XEnumVARIANT::Skip - Skip over celt elements in the collection.
//*********************************************************************
STDMETHODIMP CLitPropertyPages::XEnumVARIANT::Skip(unsigned long celt)
{

    HRESULT             hr ;
    ULONG               l  ;
    CString             tmpCString ;
    CLitPropertyPage*   tmpPropPage = NULL ;

    METHOD_PROLOGUE(CLitPropertyPages, EnumVARIANT)

    // Retrieve the next celt elements.
    for (l = 0 ; pThis->m_CurrentPosition != NULL && celt != 0 ; l++)
```

```
        {
        tmpPropPage = pThis->m_PropertyPages.GetNext(pThis->m_CurrentPosition)
        ;
        celt-- ;
        }

    hr = (celt == 0 ? NOERROR : ResultFromScode( S_FALSE )) ;

    return hr ;


}

//*********************************************************************
// XEnumVARIANT::Reset -    Reset enumerator position to the beginning.
//*********************************************************************
STDMETHODIMP CLitPropertyPages::XEnumVARIANT::Reset()
{
    METHOD_PROLOGUE(CLitPropertyPages, EnumVARIANT)
    pThis->m_CurrentPosition = pThis->m_PropertyPages.GetHeadPosition() ;
    return NOERROR ;
}


//*********************************************************************
// XEnumVARIANT::Clone -    unsupported feature
//*********************************************************************
STDMETHODIMP CLitPropertyPages::XEnumVARIANT::Clone(IEnumVARIANT FAR* FAR* )
{
    return NOERROR ;
}
```

```
// mfcdual.h: Helpful macros for adding dual interface support to
//            MFC applications

// This is a part of the Microsoft Foundation Classes C++ library.
// Copyright (C) 1992-1996 Microsoft Corporation
// All rights reserved.
//
// This source code is only intended as a supplement to the
// Microsoft Foundation Classes Reference and related
// electronic documentation provided with the library.
// See these sources for detailed information regarding the
// Microsoft Foundation Classes product.

/////////////////////////////////////////////////////////////////////////////
// BEGIN_DUAL_INTERFACE_PART is just like BEGIN_INTERFACE_PART,
// except that it also adds the IDispatch methods to your class
// declaration.


#ifndef __MFCDUAL_H__
#define __MFCDUAL_H__


#define BEGIN_DUAL_INTERFACE_PART(localClass, baseClass) \
    BEGIN_INTERFACE_PART(localClass, baseClass) \
        STDMETHOD(GetTypeInfoCount)(UINT FAR* pctinfo); \
        STDMETHOD(GetTypeInfo)(UINT itinfo, LCID lcid, ITypeInfo FAR* FAR*
        pptinfo); \
        STDMETHOD(GetIDsOfNames)(REFIID riid, OLECHAR FAR* FAR* rgszNames, UINT
        cNames, LCID lcid, DISPID FAR* rgdispid); \
        STDMETHOD(Invoke)(DISPID dispidMember, REFIID riid, LCID lcid, WORD
        wFlags, DISPPARAMS FAR* pdispparams, VARIANT FAR* pvarResult, EXCEPINFO
        FAR* pexcepinfo, UINT FAR* puArgErr); \

/////////////////////////////////////////////////////////////////////////////
// END_DUAL_INTERFACE_PART is just like END_INTERFACE_PART. It
// is only added for symmetry...
#define END_DUAL_INTERFACE_PART(localClass) \
    END_INTERFACE_PART(localClass) \


/////////////////////////////////////////////////////////////////////////////
// DELEGATE_DUAL_INTERFACE expands to define the standard IDispatch
// methods for a dual interface, delegating back to the default
// MFC implementation
#define DELEGATE_DUAL_INTERFACE(objectClass, dualClass) \
    STDMETHODIMP_(ULONG) objectClass::X##dualClass::AddRef() \
    { \
        METHOD_PROLOGUE(objectClass, dualClass) \
        return pThis->ExternalAddRef(); \
    } \
    STDMETHODIMP_(ULONG) objectClass::X##dualClass::Release() \
    { \
        METHOD_PROLOGUE(objectClass, dualClass) \
        return pThis->ExternalRelease(); \
    } \
    STDMETHODIMP objectClass::X##dualClass::QueryInterface( \
        REFIID iid, LPVOID* ppvObj) \
    { \
        METHOD_PROLOGUE(objectClass, dualClass) \
        return pThis->ExternalQueryInterface(&iid, ppvObj); \
```

```
    } \
    STDMETHODIMP objectClass::X##dualClass::GetTypeInfoCount( \
        UINT FAR* pctinfo) \
    { \
        METHOD_PROLOGUE(objectClass, dualClass) \
        LPDISPATCH lpDispatch = pThis->GetIDispatch(FALSE); \
        ASSERT(lpDispatch != NULL); \
        return lpDispatch->GetTypeInfoCount(pctinfo); \
    } \
    STDMETHODIMP objectClass::X##dualClass::GetTypeInfo( \
        UINT itinfo, LCID lcid, ITypeInfo FAR* FAR* pptinfo) \
    { \
        METHOD_PROLOGUE(objectClass, dualClass) \
        LPDISPATCH lpDispatch = pThis->GetIDispatch(FALSE); \
        ASSERT(lpDispatch != NULL); \
        return lpDispatch->GetTypeInfo(itinfo, lcid, pptinfo); \
    } \
    STDMETHODIMP objectClass::X##dualClass::GetIDsOfNames( \
        REFIID riid, OLECHAR FAR* FAR* rgszNames, UINT cNames, \
        LCID lcid, DISPID FAR* rgdispid) \
    { \
        METHOD_PROLOGUE(objectClass, dualClass) \
        LPDISPATCH lpDispatch = pThis->GetIDispatch(FALSE); \
        ASSERT(lpDispatch != NULL); \
        return lpDispatch->GetIDsOfNames(riid, rgszNames, cNames, \
                                         lcid, rgdispid); \
    } \
    STDMETHODIMP objectClass::X##dualClass::Invoke( \
        DISPID dispidMember, REFIID riid, LCID lcid, WORD wFlags, \
        DISPPARAMS FAR* pdispparams, VARIANT FAR* pvarResult, \
        EXCEPINFO FAR* pexcepinfo, UINT FAR* puArgErr) \
    { \
        METHOD_PROLOGUE(objectClass, dualClass) \
        LPDISPATCH lpDispatch = pThis->GetIDispatch(FALSE); \
        ASSERT(lpDispatch != NULL); \
        return lpDispatch->Invoke(dispidMember, riid, lcid, \
                                  wFlags, pdispparams, pvarResult, \
                                  pexcepinfo, puArgErr); \
    } \


////////////////////////////////////////////////////////////////////////
// TRY_DUAL and CATCH_ALL_DUAL are used to provide exception handling
// for your dual interface methods. CATCH_ALL_DUAL takes care of
// returning the appropriate error code.

#define TRY_DUAL(iidSource) \
    HRESULT _hr = S_OK; \
    REFIID  _riidSource = iidSource; \
    TRY \

#define CATCH_ALL_DUAL \
    CATCH(COleException, e) \
    { \
        _hr = e->m_sc; \
    } \
    AND_CATCH_ALL(e) \
    { \
        AFX_MANAGE_STATE(pThis->m_pModuleState); \
        _hr = DualHandleException(_riidSource, e); \
    } \
```

```
    END_CATCH_ALL \
    return _hr; \

////////////////////////////////////////////////////////////////////////
// DualHandleException is a helper function used to set the system's
// error object, so that container applications that call through
// VTBLs can retrieve rich error information
HRESULT DualHandleException(REFIID riidSource, const CException*
pAnyException);

////////////////////////////////////////////////////////////////////////
// DECLARE_DUAL_ERRORINFO expands to declare the ISupportErrorInfo
// support class. It works together with DUAL_ERRORINFO_PART and
// IMPLEMENT_DUAL_ERRORINFO defined below.
#define DECLARE_DUAL_ERRORINFO() \
    BEGIN_INTERFACE_PART(SupportErrorInfo, ISupportErrorInfo) \
        STDMETHOD(InterfaceSupportsErrorInfo)(THIS_ REFIID riid); \
    END_INTERFACE_PART(SupportErrorInfo) \

////////////////////////////////////////////////////////////////////////
// DUAL_ERRORINFO_PART adds the appropriate entry to the interface map
// for ISupportErrorInfo, if you used DECLARE_DUAL_ERRORINFO.
#define DUAL_ERRORINFO_PART(objectClass) \
    INTERFACE_PART(objectClass, IID_ISupportErrorInfo, SupportErrorInfo) \

////////////////////////////////////////////////////////////////////////
// IMPLEMENT_DUAL_ERRORINFO expands to an implementation of
// ISupportErrorInfo which matches the declaration in
// DECLARE_DUAL_ERRORINFO.
#define IMPLEMENT_DUAL_ERRORINFO(objectClass, riidSource) \
    STDMETHODIMP_(ULONG) objectClass::XSupportErrorInfo::AddRef() \
    { \
        METHOD_PROLOGUE(objectClass, SupportErrorInfo) \
        return pThis->ExternalAddRef(); \
    } \
    STDMETHODIMP_(ULONG) objectClass::XSupportErrorInfo::Release() \
    { \
        METHOD_PROLOGUE(objectClass, SupportErrorInfo) \
        return pThis->ExternalRelease(); \
    } \
    STDMETHODIMP objectClass::XSupportErrorInfo::QueryInterface( \
        REFIID iid, LPVOID* ppvObj) \
    { \
        METHOD_PROLOGUE(objectClass, SupportErrorInfo) \
        return pThis->ExternalQueryInterface(&iid, ppvObj); \
    } \
    STDMETHODIMP objectClass::XSupportErrorInfo::InterfaceSupportsErrorInfo( \
        REFIID iid) \
    { \
        METHOD_PROLOGUE(objectClass, SupportErrorInfo) \
        return (iid == riidSource) ? S_OK : S_FALSE; \
    }



////////////////////////////////////////////////////////////////////////


#endif  //__MFCDUAL_H__
```

```
// mfcdual.cpp: Helpful functions for adding dual interface support to
//              MFC applications

// This is a part of the Microsoft Foundation Classes C++ library.
// Copyright (C) 1992-1996 Microsoft Corporation
// All rights reserved.
//
// This source code is only intended as a supplement to the
// Microsoft Foundation Classes Reference and related
// electronic documentation provided with the library.
// See these sources for detailed information regarding the
// Microsoft Foundation Classes product.

#include    "stdafx.h"
#include    "Control.h"
#include    "Controls.h"
#include    "LitPropPage.h"
#include    "PropPages.h"
#include    "Template.h"

#include <afxpriv.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// DualHandleException

HRESULT DualHandleException(REFIID riidSource, const CException*
pAnyException)
{
    USES_CONVERSION;

    ASSERT_VALID(pAnyException);

    TRACE0("DualHandleException called\n");

    // Set ErrInfo object so that VTLB binding container
    // applications can get rich error information.
    ICreateErrorInfo* pcerrinfo;
    HRESULT hr = CreateErrorInfo(&pcerrinfo);
    if (SUCCEEDED(hr))
    {
        TCHAR   szDescription[256];
        LPCTSTR pszDescription = szDescription;
        GUID    guid = GUID_NULL;
        DWORD   dwHelpContext = 0;
        BSTR    bstrHelpFile = NULL;
        BSTR    bstrSource = NULL;
        if (pAnyException->IsKindOf(RUNTIME_CLASS(COleDispatchException)))
        {
            // specific IDispatch style exception
            COleDispatchException* e = (COleDispatchException*)pAnyException;

            guid = riidSource;
            hr = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
                              (e->m_wCode + 0x200));

            pszDescription = e->m_strDescription;
```

```
            dwHelpContext = e->m_dwHelpContext;

            // propagate source and help file if present
            // call ::SysAllocString directly so no further exceptions are
            thrown
            if (!e->m_strHelpFile.IsEmpty())
                bstrHelpFile = ::SysAllocString(T2COLE(e->m_strHelpFile));
            if (!e->m_strSource.IsEmpty())
                bstrSource = ::SysAllocString(T2COLE(e->m_strSource));

        }
        else if (pAnyException->IsKindOf(RUNTIME_CLASS(CMemoryException)))
        {
            // failed memory allocation
            AfxLoadString(AFX_IDP_FAILED_MEMORY_ALLOC, szDescription);
            hr = E_OUTOFMEMORY;
        }
        else
        {
            // other unknown/uncommon error
            AfxLoadString(AFX_IDP_INTERNAL_FAILURE, szDescription);
            hr = E_UNEXPECTED;
        }

        if (bstrHelpFile == NULL && dwHelpContext != 0)
            bstrHelpFile =
            ::SysAllocString(T2COLE(AfxGetApp()->m_pszHelpFilePath));

        if (bstrSource == NULL)
            bstrSource = ::SysAllocString(T2COLE(AfxGetAppName()));

        // Set up ErrInfo object
        pcerrinfo->SetGUID(guid);
        pcerrinfo->SetDescription(::SysAllocString(T2COLE(pszDescription)));
        pcerrinfo->SetHelpContext(dwHelpContext);
        pcerrinfo->SetHelpFile(bstrHelpFile);
        pcerrinfo->SetSource(bstrSource);

        TRACE("\tSource = %ws\n", bstrSource);
        TRACE("\tDescription = %s\n", pszDescription);
        TRACE("\tHelpContext = %lx\n", dwHelpContext);
        TRACE("\tHelpFile = %ws\n", bstrHelpFile);

        // Set the ErrInfo object for the current thread
        IErrorInfo* perrinfo;
        if (SUCCEEDED(pcerrinfo->QueryInterface(IID_IErrorInfo,
        (LPVOID*)&perrinfo)))
        {
            SetErrorInfo(0, perrinfo);
            perrinfo->Release();
        }

        pcerrinfo->Release();
    }

    TRACE("DualHandleException returning HRESULT %lx\n", hr);

    return hr;
}
```

```
////////////////////////////////////////////////////////////////////////////
//
// LitTemplate.h:   main header file for the LITTEMPLATE DLL
// Abstract:        This class implements the Control object.
//
//  Date      By    Comments
// -------  ------  --------
// 13sep96    npt   Initial Version
//
////////////////////////////////////////////////////////////////////////////

#ifndef __LITTEMPLATE_H__
#define __LITTEMPLATE_H__


#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols

////////////////////////////////////////////////////////////////////////////
// CLitTemplateApp
// See LitTemplate.cpp for the implementation of this class
//

class CLitTemplateApp : public CWinApp
{
public:
    CLitTemplateApp();

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CLitTemplateApp)
    public:
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL

    //{{AFX_MSG(CLitTemplateApp)
        // NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};


////////////////////////////////////////////////////////////////////////////

#endif  //__LITTEMPLATE_H__
```

```
////////////////////////////////////////////////////////////////////////////
//
// LitTemplate.cpp: Defines the initialization routines for the DLL.
// Abstract:        This class implements the Control object.
//
//  Date      By    Comments
// -------  ------  --------
// 13sep96    npt   Initial Version
//
////////////////////////////////////////////////////////////////////////////

#include    "stdafx.h"
#include    "LitTemplate.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////////////////////////////////////////////
// CLitTemplateApp

BEGIN_MESSAGE_MAP(CLitTemplateApp, CWinApp)
    //{{AFX_MSG_MAP(CLitTemplateApp)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
// CLitTemplateApp construction

CLitTemplateApp::CLitTemplateApp()
{
    // TODO: add construction code here,
    // Place all significant initialization in InitInstance
}

////////////////////////////////////////////////////////////////////////////
// The one and only CLitTemplateApp object

CLitTemplateApp theApp;

////////////////////////////////////////////////////////////////////////////
// CLitTemplateApp initialization

BOOL CLitTemplateApp::InitInstance()
{
    // Register all OLE server (factories) as running.  This enables the
    //  OLE libraries to create objects from other applications.
    COleObjectFactory::RegisterAll();

// 06sep95  npt  Start Dual interface support
//  AfxOleRegisterTypeLib(AfxGetInstanceHandle(), LIBID_LitTemplate,
_T("LitTemplate.TLB")) ;
    AfxOleRegisterTypeLib(AfxGetInstanceHandle(), LIBID_LicensIt,
     _T("LitTemplate.TLB")) ;
// End Dual interface support


    return TRUE;
```

```
}

////////////////////////////////////////////////////////////////////////////
// Special entry points required for inproc servers

STDAPI DllGetClassObject(REFCLSID rclsid, REFIID riid, LPVOID* ppv)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    return AfxDllGetClassObject(rclsid, riid, ppv);
}

STDAPI DllCanUnloadNow(void)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    return AfxDllCanUnloadNow();
}

// by exporting DllRegisterServer, you can use regsvr.exe
STDAPI DllRegisterServer(void)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    COleObjectFactory::UpdateRegistryAll();
    return S_OK;
}
```

```
////////////////////////////////////////////////////////////////////////////
//
// LitPropPage.h:   header file
// Abstract:        This class implements the LicensIt Property Page object.
//
//  Date      By    Comments
// -------  ------  --------
// 13sep96    npt   Initial Version
//
////////////////////////////////////////////////////////////////////////////

#ifndef __LITPROPPAGE_H__
#define __LITPROPPAGE_H__


/////////////////////////////////////////////////////////////////////////////
// CLitPropertyPage command target

class CLitPropertyPage : public CCmdTarget
{
    DECLARE_DYNCREATE(CLitPropertyPage)

public:
    CLitPropertyPage();           // protected constructor used by dynamic
    creation
    CLitPropertyPage(const CString& Name) ;

// Attributes
public:
    CControls   m_CControls ;   // collection of Controls on this page
    CString     key ;           // page key

private:
    DWORD       m_Version ;     // Property Page version

// Operations


public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CLitPropertyPage)
    public:
    virtual void OnFinalRelease();
    //}}AFX_VIRTUAL


// Implementation
public:
    virtual ~CLitPropertyPage();
    BOOL    Close(LPSTORAGE pStg) ;
    BOOL    Open (LPSTORAGE pStg) ;

protected:
    // Generated message map functions
    //{{AFX_MSG(CLitPropertyPage)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG

    DECLARE_MESSAGE_MAP()
```

```
    // Generated OLE dispatch map functions
    //{{AFX_DISPATCH(CLitPropertyPage)
    CString m_PageName;
    afx_msg void OnNameChanged();
    LPDISPATCH m_controls;
    afx_msg void OnControlsChanged();
    CString m_oID;
    afx_msg void OnOIDChanged();
    //}}AFX_DISPATCH

protected:

    DECLARE_DISPATCH_MAP()
    DECLARE_INTERFACE_MAP()

// 25sep95  npt  Start Dual interface support
    BEGIN_DUAL_INTERFACE_PART(DualLitPropertyPage, IDualLitPropertyPage)
        STDMETHOD(put_Name)(THIS_ VARIANT newName);
        STDMETHOD(get_Name)(THIS_ VARIANT FAR* retval);
        STDMETHOD(Controls)(THIS_ VARIANT FAR* retval);
        STDMETHOD(put_OID)(THIS_ VARIANT newOID);
        STDMETHOD(get_OID)(THIS_ VARIANT FAR* retval);
    END_DUAL_INTERFACE_PART(DualLitPropertyPage)
// End Dual interface support

    //      add declaration of ISupportErrorInfo implementation
    //      to indicate we support the OLE Automation error object
    DECLARE_DUAL_ERRORINFO()

};

/////////////////////////////////////////////////////////////////////////////

#endif  //__LITPROPPAGE_H__
```

```
////////////////////////////////////////////////////////////////////////////
//
// LitPropPage.cpp: implementation file
// Abstract:        This class implements the LicensIt Property Page object.
//
//  Date      By    Comments
// -------  ------  --------
// 13sep96    npt   Initial Version
//
////////////////////////////////////////////////////////////////////////////

#include    "stdafx.h"
#include    "littemplate.h"
#include    "Control.h"
#include    "Controls.h"
#include    "LitPropPage.h"


#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__ ;
#endif

////////////////////////////////////////////////////////////////////////////
// CLitPropertyPage

IMPLEMENT_DYNCREATE(CLitPropertyPage, CCmdTarget)

CLitPropertyPage::CLitPropertyPage()
{
    EnableAutomation();
    m_Version = VERSION1 ;
    m_PageName = "" ;
    m_oID = "" ;
    key = "" ;

// Record IDispatch for CControls (Controls collection)
    m_controls = m_CControls.GetIDispatch(FALSE) ;
}

CLitPropertyPage::CLitPropertyPage(const CString& Name)
{
    EnableAutomation();

    m_Version  = VERSION1 ;
//  m_PageName = Name ;
    m_oID      = Name ;
    key        = Name ;

// Record IDispatch for CControls (Controls collection)
    m_controls = m_CControls.GetIDispatch(FALSE) ;

}

CLitPropertyPage::~CLitPropertyPage()
{

}


void CLitPropertyPage::OnFinalRelease()
```

```
{
    // When the last reference for an automation object is released
    // OnFinalRelease is called.  The base class will automatically
    // deletes the object.  Add additional cleanup required for your
    // object before calling the base class.

    CCmdTarget::OnFinalRelease();
}


BEGIN_MESSAGE_MAP(CLitPropertyPage, CCmdTarget)
    //{{AFX_MSG_MAP(CLitPropertyPage)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_DISPATCH_MAP(CLitPropertyPage, CCmdTarget)
    //{{AFX_DISPATCH_MAP(CLitPropertyPage)
    DISP_PROPERTY_NOTIFY(CLitPropertyPage, "Name", m_PageName, OnNameChanged,
    VT_BSTR)
    DISP_PROPERTY_NOTIFY(CLitPropertyPage, "Controls", m_controls,
    OnControlsChanged, VT_DISPATCH)
    DISP_PROPERTY_NOTIFY(CLitPropertyPage, "OID", m_oID, OnOIDChanged,
    VT_BSTR)
    //}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()

// Note: we add support for IID_ILitPropertyPage to support typesafe binding
//  from VBA.  This IID must match the GUID that is attached to the
//  dispinterface in the .ODL file.

BEGIN_INTERFACE_MAP(CLitPropertyPage, CCmdTarget)
    INTERFACE_PART(CLitPropertyPage, DIID_ILitPropertyPage, Dispatch)
// 05sep95  npt  Start Dual interface support
    INTERFACE_PART(CLitPropertyPage, IID_IDualLitPropertyPage,
    DualLitPropertyPage)
    DUAL_ERRORINFO_PART(CLitPropertyPage)
// End Dual interface support
END_INTERFACE_MAP()


//************************************************************************
// Close -  Write the data for the Property Page to the storage passed
//          in.
//************************************************************************
BOOL CLitPropertyPage::Close(LPSTORAGE pStg)
{

// Store the data for the property page in this storage. Create a new storage
// to store each of the controls in.
    USES_CONVERSION;
    LPSTORAGE       pStgControls = NULL ;
    LPSTREAM        pStream = NULL ;
//  char    far*    szStreamName  = "PageName" ;
//  char    far*    szStorageName = "Controls" ;
    char            buf[80] ;
    ULONG           written ;
    HRESULT         hr ;


// Get the string for the "PageName" stream
    LoadString(AfxGetInstanceHandle(), IDS_STREAM_PAGE_NAME, (LPTSTR)&buf, 80)
```

```
    ;
//  VERIFY(pStg->CreateStream(T2OLE(szStreamName),
        VERIFY(pStg->CreateStream(T2OLE(buf),
                                  STGM_CREATE | STGM_READWRITE |
                STGM_SHARE_EXCLUSIVE,
                                  0, 0, &pStream ) == S_OK);
        ASSERT(pStream != NULL) ;


// Write the version number of this property page to the storage stream.
        hr = pStream->Write((DWORD *)&m_Version, sizeof(DWORD), (ULONG
    *)&written) ;
        WriteString(pStream, (LPCTSTR)m_PageName) ;
        WriteString(pStream, (LPCTSTR)m_oID) ;

        pStream->Release() ;

// Create a storage for the controls on this page.
        LoadString(AfxGetInstanceHandle(), IDS_STORAGE_CONTROLS, (LPTSTR)&buf,
    80) ;
//  VERIFY(pStg->CreateStorage(T2OLE(szStorageName),
        VERIFY(pStg->CreateStorage(T2OLE(buf),
                STGM_CREATE | STGM_READWRITE | STGM_SHARE_EXCLUSIVE,
                0, 0, &pStgControls) == S_OK) ;
        m_CControls.Close(pStgControls) ;
        pStgControls->Release() ;

        return TRUE ;
}

//*****************************************************************************
// Open -   Read the data for the Property Page from the passed in
//          storage
//*****************************************************************************
BOOL CLitPropertyPage::Open(LPSTORAGE pStg)
{

        USES_CONVERSION;
// Read the Page name stream from the storage file
        LPSTORAGE       pStgControls = NULL ;
        LPSTREAM        pStream = NULL ;
        char            buf[80] ;
        DWORD           version ;


// Get the string for the "PageName" stream
        LoadString(AfxGetInstanceHandle(), IDS_STREAM_PAGE_NAME, (LPTSTR)&buf,
    80) ;
        VERIFY(pStg->OpenStream(T2OLE(buf), NULL,
                                  STGM_READ | STGM_SHARE_EXCLUSIVE, 0L, &pStream
                ) == S_OK) ;
        ASSERT(pStream != NULL) ;

// Read the version number of this property page to the storage stream.
        pStream->Read((DWORD *)&version,   sizeof(DWORD), NULL) ;
        m_Version = version ;

// Read the page name
        ReadString(pStream, (LPTSTR)&buf) ;
        key = buf ;
        m_PageName = buf ;
```

```
// Read the OID
    ReadString(pStream, (LPTSTR)&buf) ;
    m_oID = buf ;
    key   = m_oID ;
    pStream->Release() ;

    // Read the Controls
    LoadString(AfxGetInstanceHandle(), IDS_STORAGE_CONTROLS, (LPTSTR)&buf, 80)
    ;
    VERIFY(pStg->OpenStorage(T2OLE(buf), NULL,
            STGM_READ | STGM_SHARE_EXCLUSIVE, NULL, 0L, &pStgControls ) ==
            S_OK) ;
    ASSERT(pStgControls != NULL) ;

    // Read each of the property pages
    m_CControls.Open(pStgControls) ;

    pStgControls->Release() ;


    return TRUE ;
}


/////////////////////////////////////////////////////////////////////////////
// CLitPropertyPage message handlers

void CLitPropertyPage::OnNameChanged()
{
    // TODO: Add notification handler code

}

void CLitPropertyPage::OnControlsChanged()
{
    // TODO: Add notification handler code

}

void CLitPropertyPage::OnOIDChanged()
{
    // TODO: Add notification handler code

}

/////////////////////////////////////////////////////////////////////////////
// DUAL INTERFACE FUNCTIONS
/////////////////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////////////
// delegate standard IDispatch methods to MFC IDispatch implementation
// Macro defined in MFCDUAL.H
DELEGATE_DUAL_INTERFACE(CLitPropertyPage, DualLitPropertyPage)

// Implement ISupportErrorInfo to indicate we support the
// OLE Automation error handler.
IMPLEMENT_DUAL_ERRORINFO(CLitPropertyPage, IID_IDualLitPropertyPage)


//****************************************************************************
// put_Name -
//****************************************************************************
```

```
STDMETHODIMP CLitPropertyPage::XDualLitPropertyPage::put_Name(VARIANT newName)
{
    METHOD_PROLOGUE(CLitPropertyPage, DualLitPropertyPage)
    TRY_DUAL(IID_IDualLitPropertyPage)
        {
        if (newName.vt == VT_BSTR)
            {
            pThis->m_PageName = newName.bstrVal ;
            }
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*************************************************************************
// get_Name -
//*************************************************************************
STDMETHODIMP CLitPropertyPage::XDualLitPropertyPage::get_Name(VARIANT FAR*
retval)
{
    METHOD_PROLOGUE(CLitPropertyPage, DualLitPropertyPage)
    TRY_DUAL(IID_IDualLitPropertyPage)
        {
        VARIANT     m_vaTemp ;
        ::VariantInit(&m_vaTemp); // necessary initialization
        m_vaTemp.vt = VT_BSTR ;
        m_vaTemp.bstrVal = pThis->m_PageName.AllocSysString() ;
        *retval = m_vaTemp ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*************************************************************************
// Controls -
//*************************************************************************
STDMETHODIMP CLitPropertyPage::XDualLitPropertyPage::Controls(VARIANT FAR*
retval)
{
    METHOD_PROLOGUE(CLitPropertyPage, DualLitPropertyPage)
    TRY_DUAL(IID_IDualLitPropertyPage)
        {
        VARIANT     m_vaTemp ;
        ::VariantInit(&m_vaTemp); // necessary initialization
        m_vaTemp.vt       = VT_DISPATCH ;
        m_vaTemp.pdispVal = pThis->m_CControls.GetIDispatch(TRUE) ;
        *retval = m_vaTemp ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}


//*************************************************************************
// put_OID -
//*************************************************************************
STDMETHODIMP CLitPropertyPage::XDualLitPropertyPage::put_OID(VARIANT newOID)
{
    METHOD_PROLOGUE(CLitPropertyPage, DualLitPropertyPage)
    TRY_DUAL(IID_IDualLitPropertyPage)
        {
        if (newOID.vt == VT_BSTR)
```

```
            {
            pThis->m_oID = newOID.bstrVal ;
            }
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*************************************************************************
// get_OID -
//*************************************************************************
STDMETHODIMP CLitPropertyPage::XDualLitPropertyPage::get_OID(VARIANT FAR*
retval)
{
    METHOD_PROLOGUE(CLitPropertyPage, DualLitPropertyPage)
    TRY_DUAL(IID_IDualLitPropertyPage)
        {
        VARIANT     m_vaTemp ;
        ::VariantInit(&m_vaTemp); // necessary initialization
        m_vaTemp.vt = VT_BSTR ;
        m_vaTemp.bstrVal = pThis->m_oID.AllocSysString() ;
        *retval = m_vaTemp ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}
```

```
//////////////////////////////////////////////////////////////////////////////
//
// Filename:    initiids.h
// Abstract:    This source file includes TPLDual.h (definitions of dual
interface
//              classes and guids) and global helper functions.
//
//  Date      By    Comments
// -------  ------  --------
// 13sep96    npt   Initial Version
//
//////////////////////////////////////////////////////////////////////////////

#ifndef __INITIIDS_H__
#define __INITIIDS_H__


#define    VERSION1        0x00000001L


long WriteString(LPSTREAM pStream, LPCTSTR pSz) ;
long ReadString (LPSTREAM pStream, LPTSTR  pSz) ;

//////////////////////////////////////////////////////////////////////////////

#endif  //__INITIIDS_H__
```

```
////////////////////////////////////////////////////////////////////////////
//
// INITIIDS.CPP
// Abstract:        defines IIDs for automation objects
//
//  Date      By    Comments
// -------  ------  --------
// 13sep96    npt   Initial Version
//
////////////////////////////////////////////////////////////////////////////

// This must NOT be built with the precompiled header

#include    <afxwin.h>
#include    <ole2.h>
#include    <initguid.h>
#include    "TPLDual.h"     // header file generated by MKTYPLIB.EXE
#include    "resource.h"


//**************************************************************************
// WriteString -    global helper function to write a string to an OLE
//                  stream. Strings are written as a length (long)
//                  followed by the string data. If a null string,
//                  the length (4 as a long) is followed by the word
//                  NULL.
//  return:         long - number of bytes written.
//**************************************************************************
long WriteString(LPSTREAM pStream, LPCTSTR pSz)
{
//  char    *szNullStr = "NULL" ;
    char    buf[80] ;
    LPSTR   pStr ;
    long    written ;

    long    len = lstrlen(pSz) ;
    if (len == 0)
        {
        LoadString(AfxGetInstanceHandle(), IDS_NULL_STRING, (LPTSTR)&buf, 80)
        ;
        len = lstrlen(buf) ;
        pStr = buf ;
        }
    else
        {
        pStr = (LPSTR)pSz ;
        }

// Write string length
    pStream->Write((long *)&len,   sizeof(long), NULL) ;
// Now write the string
    pStream->Write(pStr, len, (ULONG *)&written) ;

    return written ;
}


//**************************************************************************
// ReadString -     global helper function to read a string from an OLE
//                  stream. Strings are written as a length (long)
//                  followed by the string data. If a null string,
//                  the length (4 as a long) is followed by the word
```

```
//                  NULL.
//  return:         long - number of bytes written.
//*****************************************************************************
long ReadString(LPSTREAM pStream, LPTSTR pSz)
{
    char    buf[80] ;
    long    read ;
    long    count ;

// Read string length
    pStream->Read((long *)&count,   sizeof(long), (ULONG *)&read) ;
// Now read the actual string
    if (count > 0)
        {
        pStream->Read((LPTSTR)pSz, count, (ULONG *)&read) ;
        LoadString(AfxGetInstanceHandle(), IDS_NULL_STRING, (LPTSTR)&buf, 80)
        ;
        // Is this a NULL string?
        if (lstrcmp(buf, pSz) == 0)
            {
            count = 0L ;        // if a null string was read ("NULL"), return
            *pSz = '\0' ;       // a null string.
            }
        else
            *(pSz + count) = '\0' ; // terminate the string with a null
        }

    return read ;
}
```

```
////////////////////////////////////////////////////////////////////////////
//
// Filename:    Controls.h
// Abstract:    This class implements the collection of Controls on a Property
Page
//
//  Date       By    Comments
// -------  ------  --------
// 16sep96    npt   Initial Version
//
////////////////////////////////////////////////////////////////////////////

#ifndef __CONTROLS_H__
#define __CONTROLS_H__

// Map (Dictionary collection) to hold controls for a given property page.
typedef CMap<CString, LPCSTR, CControl*, CControl*> CControlMap ;
typedef CList<CControl*, CControl*> CControlList ;

// special versions for CControls
//void AFXAPI ConstructElements(CControl* pControls, int nCount);
//void AFXAPI DestructElements(CControl*  pControls, int nCount);


/////////////////////////////////////////////////////////////////////////////
// CControls command target

class CControls : public CCmdTarget
{
    DECLARE_DYNCREATE(CControls)

    CControls();            // protected constructor used by dynamic creation

// Attributes
public:
    CControlList    m_Controls ;    // Collection of controls on this page
    POSITION        m_CurrentPosition ;

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CControls)
    public:
    virtual void OnFinalRelease();
    //}}AFX_VIRTUAL
    BOOL        Close(LPSTORAGE pStg) ;
    BOOL        Open (LPSTORAGE pStg) ;
    POSITION    Lookup(LPCSTR key) ;
    void        AddtoList(CControl* newControl) ;


// Implementation
protected:
public:
    virtual ~CControls();

    // Generated message map functions
    //{{AFX_MSG(CControls)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG
```

```
    DECLARE_MESSAGE_MAP()
    // Generated OLE dispatch map functions
    //{{AFX_DISPATCH(CControls)
    afx_msg LPDISPATCH GetItem(const VARIANT FAR& item);
    afx_msg long Count();
    afx_msg LPDISPATCH Add(LPCTSTR key);
    afx_msg VARIANT Remove(const VARIANT FAR& item);
    //}}AFX_DISPATCH
    DECLARE_DISPATCH_MAP()
    afx_msg LPUNKNOWN _NewEnum(void);

// Create an enumerator class to support the automation collection
    BEGIN_INTERFACE_PART(EnumVARIANT, IEnumVARIANT)
        STDMETHOD(Next)(THIS_ unsigned long celt, VARIANT FAR* rgvar, unsigned
        long FAR* pceltFetched) ;
        STDMETHOD(Skip)(THIS_ unsigned long celt) ;
        STDMETHOD(Reset)(THIS) ;
        STDMETHOD(Clone)(THIS_ IEnumVARIANT FAR* FAR* ppenum) ;
        XEnumVARIANT() ;
    END_INTERFACE_PART(EnumVARIANT)


    DECLARE_INTERFACE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

#endif  //__CONTROLS_H__
```

```
////////////////////////////////////////////////////////////////////////////
//
// Filename:    Controls.cpp
// Abstract:    This class implements the collection of Controls on a Property
Page
//
//  Date      By    Comments
// -------  ------  --------
// 16sep96    npt   Initial Version
//
////////////////////////////////////////////////////////////////////////////

#include    "stdafx.h"
#include    "littemplate.h"
#include    "Control.h"
#include    "Controls.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////////////////////////////////////////////
// CControls
IMPLEMENT_DYNCREATE(CControls, CCmdTarget)

CControls::CControls()
{
    EnableAutomation();
}

CControls::~CControls()
{
// Make sure there all reference counts are zero.
    long    Count = m_Controls.GetCount() ;
    if (Count > 0)
        {
        POSITION pos = m_Controls.GetHeadPosition();
        while( pos != NULL )
            {
            CControl*       pControl ;
            pControl = m_Controls.GetNext(pos);
            while (pControl->ExternalRelease())
                ;
            }
        }

// If we have any Controls in our collection, remove 'em
    m_Controls.RemoveAll() ;

}

//*****************************************************************************
// Close -  Write each of the controls to the storage passed in.
//*****************************************************************************
BOOL CControls::Close(LPSTORAGE pStg)
{

// Let each control create a new stream in the storage to write itself to.
    POSITION    pos = m_Controls.GetHeadPosition();
    while( pos != NULL )
```

```
        {
        CControl*       pControl ;
        CString         string ;

        pControl = m_Controls.GetNext(pos) ;
        pControl->AssertValid() ;
        pControl->Close(pStg) ;
        }

    return TRUE ;
}


//*****************************************************************************
// Open -   Read the Controls from the OLE storage
//*****************************************************************************
BOOL CControls::Open(LPSTORAGE pStg)
{

// Iterate through the controls in the storage
    USES_CONVERSION;
    LPSTREAM            pStreamControl = NULL ;
    LPENUMSTATSTG       pEnum = NULL ;
    LPMALLOC            pMalloc = NULL ;
    STATSTG             statstg ;
    CControl*           newControl ;

    ::CoGetMalloc(MEMCTX_TASK, &pMalloc) ;  // Assumes AfxOleInit was called
    VERIFY(pStg->EnumElements(0, NULL, 0, &pEnum) == S_OK) ;
    while (pEnum->Next(1, &statstg, NULL) == S_OK)
        {
        if (statstg.type == STGTY_STREAM)
            {
            VERIFY(pStg->OpenStream(statstg.pwcsName, NULL,
                                    STGM_READ | STGM_SHARE_EXCLUSIVE,
                                    0, &pStreamControl) == S_OK) ;
            ASSERT(pStreamControl != NULL) ;
            newControl = new CControl ;

            newControl->Open(pStreamControl) ;

            AddtoList(newControl) ;     // Add the control to the list
            pStreamControl->Release() ;
            pMalloc->Free(statstg.pwcsName) ;   // avoids memory leaks
            }
        }
    pMalloc->Release() ;
    pEnum->Release() ;
    return TRUE ;
}


//*****************************************************************************
// AddtoList -  Add the passed in Control to the collection
//*****************************************************************************
void    CControls::AddtoList(CControl* newControl)
{
    CString             tmp = newControl->key ;
    POSITION            pos = Lookup((LPCSTR)tmp) ;
    LPDISPATCH          lpDisp = NULL ;

    if ( pos == NULL )
```

```
        {
        m_Controls.AddTail(newControl) ;
        lpDisp = newControl->GetIDispatch(TRUE) ;    // AddRef
        }
    return ;
}


void CControls::OnFinalRelease()
{
    // When the last reference for an automation object is released
    // OnFinalRelease is called.  The base class will automatically
    // deletes the object.  Add additional cleanup required for your
    // object before calling the base class.

    CCmdTarget::OnFinalRelease();
}


BEGIN_MESSAGE_MAP(CControls, CCmdTarget)
    //{{AFX_MSG_MAP(CControls)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_DISPATCH_MAP(CControls, CCmdTarget)
    //{{AFX_DISPATCH_MAP(CControls)
    DISP_FUNCTION(CControls, "Item", GetItem, VT_DISPATCH, VTS_VARIANT)
    DISP_DEFVALUE(CControls, "Item")
    DISP_FUNCTION(CControls, "Count", Count, VT_I4, VTS_NONE)
    DISP_FUNCTION(CControls, "Add", Add, VT_DISPATCH, VTS_BSTR)
    DISP_FUNCTION(CControls, "Remove", Remove, VT_VARIANT, VTS_VARIANT)
    //}}AFX_DISPATCH_MAP
    DISP_PROPERTY_EX_ID(CControls, "_NewEnum", DISPID_NEWENUM, _NewEnum,
    SetNotSupported, VT_UNKNOWN)
END_DISPATCH_MAP()

// Note: we add support for IID_IControls to support typesafe binding
//  from VBA.  This IID must match the GUID that is attached to the
//  dispinterface in the .ODL file.

BEGIN_INTERFACE_MAP(CControls, CCmdTarget)
    INTERFACE_PART(CControls, DIID_IControls, Dispatch)
    INTERFACE_PART(CControls, IID_IEnumVARIANT, EnumVARIANT)
END_INTERFACE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CControls message handlers

//*****************************************************************************
// _NewEnum -   Locate and return the IUnknown for the enumerator class.
//*****************************************************************************
LPUNKNOWN CControls::_NewEnum(void)
{

    LPUNKNOWN   punkEnumVariant = NULL ;

    m_xEnumVARIANT.QueryInterface(IID_IUnknown, (LPVOID *)&punkEnumVariant) ;
    m_CurrentPosition = m_Controls.GetHeadPosition() ;

    return punkEnumVariant;
```

```
}


//*****************************************************************************
// GetItem -    Locate a CControl object in our collection and return
//              an IDispatch Interface pointer if found.
//  Args:       The input argument is a VARIANT specifying either the key
//              name of the desired control or an index number of a control.
//  Return:     NULL if entry is not in the collection (either the name was
//              not found or the index is out of bounds).
//              IDispatch pointer when the item is found.
//*****************************************************************************
LPDISPATCH CControls::GetItem(const VARIANT FAR& item)
{

    LPDISPATCH      lpDisp = NULL ;
    CString         tmpCString ;
    CControl*       tmpControl = NULL ;
    POSITION        pos = NULL ;

    if (item.vt == VT_BSTR)
        {

        CString     key = item.bstrVal ;

        pos = Lookup((LPCSTR)key) ;
        if (pos != NULL)
            {
            tmpControl = m_Controls.GetAt(pos) ;
            ASSERT(tmpControl != NULL) ;
            lpDisp = tmpControl->GetIDispatch(TRUE) ;        // AddRef
            }
        }
    else
        {
        // coerce to VT_I4
        VARIANT     va ;
        VariantInit( &va ) ;
        if (SUCCEEDED(VariantChangeType( &va, (VARIANT FAR*)&item, 0, VT_I4
        )))
            {
            long    Counter = va.lVal ;
            if (Counter <= (long)m_Controls.GetCount())
                {
                pos = m_Controls.GetHeadPosition();
                while(( pos != NULL ) && (Counter > 0))
                    {
                    tmpControl = m_Controls.GetNext(pos) ;
                    Counter-- ;
                    if (Counter == 0)
                        {
                        lpDisp = tmpControl->GetIDispatch(TRUE) ;        //
                        AddRef
                        }
                    }
                }
            }
        }

    // BUGBUG:  Implement dispatch exception if lpDisp == NULL
    return lpDisp ;
```

```
}

//************************************************************************
// Lookup - Lookup the PropertyPage with the key in the m_PropertyPages
//          collection.
//  Return: POSITION in the CList if found, NULL otherwise.
//************************************************************************
POSITION CControls::Lookup(LPCSTR key)
{
    POSITION    retPos = NULL ;
    POSITION    pos    = m_Controls.GetHeadPosition();
    POSITION    tmp ;

    while( pos != NULL )
        {
        tmp = pos ;
        CControl*   pControl = m_Controls.GetNext(pos) ;
        pControl->AssertValid() ;
        if (lstrcmp(key, pControl->key) == 0)
            {
            retPos = tmp ;
            pos = NULL ;
            }
        }

    return retPos ;
}


//************************************************************************
// Count -  Return the number of Controls in the collection.
//************************************************************************
long CControls::Count()
{

    return (long)m_Controls.GetCount() ;

}

//************************************************************************
// Add -    Add a new CControl to the dictionary using "key" as the
//          index.
//  Return: NULL if an entry already exists with the specified key.
//          IDispatch pointer on successful addition.
//************************************************************************
LPDISPATCH CControls::Add(LPCTSTR key)
{
    LPDISPATCH  lpDisp = NULL ;
    CString     tmp = key ;
    POSITION    pos = Lookup((LPCSTR)tmp) ;

    if (pos == NULL)
        {
        CControl*       newControl = new CControl() ;
        newControl->key = tmp ;
        newControl->SetOID() ;

        m_Controls.AddTail(newControl) ;
        lpDisp = newControl->GetIDispatch(TRUE) ;
        }
    return lpDisp ;
```

```
}

//*****************************************************************************
// Remove - Remove the CControl indicated by the input VARIANT.
//          (Remove using either the key name or an index).
//  Return: VT_EMPTY VARIANT.
//*****************************************************************************
VARIANT CControls::Remove(const VARIANT FAR& item)
{
    CString     tmpCString ;
    CControl*   tmpControl = NULL ;
    POSITION    pos ;

    if (item.vt == VT_BSTR)
        {

        CString     key = item.bstrVal ;
        pos = Lookup((LPCSTR)key) ;
        if (pos != NULL)
            {
            tmpControl = m_Controls.GetAt(pos) ;
            m_Controls.RemoveAt(pos) ;
            // Remove all references to the object so it gets destroyed
            while (tmpControl->ExternalRelease())
                ;
            }
        }
    else
        {
        // coerce to VT_I4
        VARIANT     va ;
        VariantInit( &va ) ;
        if (SUCCEEDED(VariantChangeType( &va, (VARIANT FAR*)&item, 0, VT_I4
        )))
            {
            long    Counter = va.lVal ;
            if (Counter <= (long)m_Controls.GetCount())
                {
                pos = m_Controls.GetHeadPosition();
                while(( pos != NULL ) && (Counter > 0))
                    {
                    POSITION    tmp = pos ;
                    tmpControl = m_Controls.GetNext(pos);
                    Counter-- ;
                    if (Counter == 0)
                        {
                        m_Controls.RemoveAt(tmp) ;
                        // Remove all references to the object so it gets
                        destroyed
                        while (tmpControl->ExternalRelease())
                            ;
                        }

                    }
                }
            }
        }

    VARIANT     vaResult ;
    VariantInit(&vaResult) ;
    vaResult.vt = VT_EMPTY ;
    return vaResult ;
```

```
}

//*****************************************************************************
//
// enumerator class XEnumVARIANT
//
//*****************************************************************************

CControls::XEnumVARIANT::XEnumVARIANT()
{
    METHOD_PROLOGUE(CControls, EnumVARIANT)

}

ULONG FAR EXPORT CControls::XEnumVARIANT::AddRef()
{
    METHOD_PROLOGUE(CControls, EnumVARIANT)
    return pThis->ExternalAddRef();
}

ULONG FAR EXPORT CControls::XEnumVARIANT::Release()
{
    METHOD_PROLOGUE(CControls, EnumVARIANT)
    return pThis->ExternalRelease();
}

HRESULT FAR EXPORT CControls::XEnumVARIANT::QueryInterface(REFIID iid, void
FAR* FAR* ppvObj)
{
    METHOD_PROLOGUE(CControls, EnumVARIANT)
    return (HRESULT)pThis->ExternalQueryInterface(&iid, ppvObj);
}

//*****************************************************************************
// XEnumVARIANT::Next - Retrieve the next element in the collection
//                      (position maintained in pThis->m_CurrentPosition).
//                      and return an IDispatch.
//*****************************************************************************
STDMETHODIMP CControls::XEnumVARIANT::Next(ULONG celt,
                        VARIANT FAR* rgvar, ULONG FAR* pceltFetched)
{
    // This sets up the "pThis" pointer so that it points to our
    // containing CControls instance
    METHOD_PROLOGUE(CControls, EnumVARIANT)

    HRESULT     hr = ResultFromScode( S_FALSE ) ;
    ULONG       l  ;
    CString     tmpCString ;
    CControl*   tmpControl = NULL ;


// pceltFetched can legally == 0
    if (pceltFetched != NULL)
        {
        *pceltFetched = 0 ;
        }
    else if (celt > 1)
        {
        return ResultFromScode( E_INVALIDARG ) ;
        }

    if (pThis->m_CurrentPosition == NULL)
```

```
        {
        return ResultFromScode( S_FALSE ) ;
        }

    for (l = 0; l < celt; l++)
        {
        VariantInit( &rgvar[l] ) ;
        }


    // Retrieve the next celt elements.
    hr = NOERROR ;
    for (l = 0 ; pThis->m_CurrentPosition != NULL && celt != 0 ; l++)
        {
        tmpControl = pThis->m_Controls.GetNext(pThis->m_CurrentPosition) ;
        celt-- ;
        rgvar[l].vt = VT_DISPATCH ;
        rgvar[l].pdispVal = tmpControl->GetIDispatch(TRUE) ; ;
        if (pceltFetched != NULL)
            (*pceltFetched)++ ;
        }

    return hr ;
}

//***************************************************************************
// XEnumVARIANT::Skip - Skip over celt elements in the collection.
//***************************************************************************
STDMETHODIMP CControls::XEnumVARIANT::Skip(unsigned long celt)
{

    HRESULT         hr ;
    ULONG           l  ;
    CString         tmpCString ;
    CControl*       tmpControl = NULL ;

    METHOD_PROLOGUE(CControls, EnumVARIANT)

    // Retrieve the next celt elements.
    for (l = 0 ; pThis->m_CurrentPosition != NULL && celt != 0 ; l++)
        {
        tmpControl = pThis->m_Controls.GetNext(pThis->m_CurrentPosition) ;
        celt-- ;
        }

    hr = (celt == 0 ? NOERROR : ResultFromScode( S_FALSE )) ;

    return hr ;


}

//***************************************************************************
// XEnumVARIANT::Reset -    Reset enumerator position to the beginning.
//***************************************************************************
STDMETHODIMP CControls::XEnumVARIANT::Reset()
{
    METHOD_PROLOGUE(CControls, EnumVARIANT)
    pThis->m_CurrentPosition = pThis->m_Controls.GetHeadPosition() ;
    return NOERROR ;
}
```

```
//*****************************************************************************
// XEnumVARIANT::Clone -    unsupported feature
//*****************************************************************************
STDMETHODIMP CControls::XEnumVARIANT::Clone(IEnumVARIANT FAR* FAR* )
{
    return NOERROR ;
}
```

```
////////////////////////////////////////////////////////////////////////////
//
// Control.h:        header file
// Abstract:         This class implements the Control object.
//
//  Date      By     Comments
// -------  ------  --------
// 13sep96    npt   Initial Version
//
////////////////////////////////////////////////////////////////////////////

#ifndef __CONTROL_H__
#define __CONTROL_H__


/////////////////////////////////////////////////////////////////////////////
// CControl command target

class CControl : public CCmdTarget
{
    DECLARE_DYNCREATE(CControl)

    CControl();           // protected constructor used by dynamic creation

// Attributes
public:
    CString     key ;

private:
    DWORD       m_Version ; // Control version

// Operations
public:
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CControl)
    public:
    virtual void OnFinalRelease();
    //}}AFX_VIRTUAL

    BOOL    Close(LPSTORAGE pStg) ;
    BOOL    Open(LPSTREAM pStream) ;
    void    SetOID(void) ;


// Implementation
public:
   virtual ~CControl();

protected:

    // Generated message map functions
    //{{AFX_MSG(CControl)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG

    DECLARE_MESSAGE_MAP()
    // Generated OLE dispatch map functions
    //{{AFX_DISPATCH(CControl)
    short m_xPos;
    afx_msg void OnXPosChanged();
    short m_yPos;
```

```
    afx_msg void OnYPosChanged();
    CString m_Name;
    afx_msg void OnNameChanged();
    CString m_Label;
    afx_msg void OnLabelChanged();
    short m_Width;
    afx_msg void OnWidthChanged();
    short m_Height;
    afx_msg void OnHeightChanged();
    CString m_oID;
    afx_msg void OnOIDChanged();
    short   m_controlType;
    afx_msg void OnControlTypeChanged();
    //}}AFX_DISPATCH
    DECLARE_DISPATCH_MAP()
    DECLARE_INTERFACE_MAP()


// 05oct95  npt  Dual interface support
    BEGIN_DUAL_INTERFACE_PART(DualControl, IDualControl)
        STDMETHOD(put_XPos)(THIS_ short newXPos);
        STDMETHOD(get_XPos)(THIS_ short* retval);
        STDMETHOD(put_YPos)(THIS_ short newYPos);
        STDMETHOD(get_YPos)(THIS_ short* retval);
        STDMETHOD(put_Name)(THIS_ BSTR newName);
        STDMETHOD(get_Name)(THIS_ BSTR* retval);
        STDMETHOD(put_Label)(THIS_ BSTR newLabel);
        STDMETHOD(get_Label)(THIS_ BSTR* retval);
        STDMETHOD(put_Width)(THIS_ short newWidth);
        STDMETHOD(get_Width)(THIS_ short* curWidth);
        STDMETHOD(put_Height)(THIS_ short newHeight);
        STDMETHOD(get_Height)(THIS_ short* curHeight);
        STDMETHOD(put_OID)(THIS_ BSTR newOID);
        STDMETHOD(get_OID)(THIS_ BSTR* retval);
        STDMETHOD(put_Type)(THIS_ short newType);
        STDMETHOD(get_Type)(THIS_ short* curType);
    END_DUAL_INTERFACE_PART(DualControl)
// End Dual interface support

    //      add declaration of ISupportErrorInfo implementation
    //      to indicate we support the OLE Automation error object
    DECLARE_DUAL_ERRORINFO()


};

/////////////////////////////////////////////////////////////////////////////

#endif  //__CONTROL_H__
```

```
////////////////////////////////////////////////////////////////////////////
//
// Control.cpp:     implementation file
// Abstract:        This class implements the Control object.
//
//  Date       By    Comments
// -------  ------  --------
// 13sep96    npt   Initial Version
//
////////////////////////////////////////////////////////////////////////////

#include    "stdafx.h"
#include    "littemplate.h"
#include    "Control.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CControl

IMPLEMENT_DYNCREATE(CControl, CCmdTarget)

CControl::CControl()
{
    EnableAutomation();

// initialize member variables to a known state.
    m_Version = VERSION1 ;
    m_xPos    = 0 ;
    m_yPos    = 0 ;
    m_Name    = "" ;
    m_Label   = "" ;
    m_oID     = "" ;
    m_Width   = 0 ;
    m_Height  = 0 ;

}

CControl::~CControl()
{
}


void CControl::OnFinalRelease()
{
    // When the last reference for an automation object is released
    // OnFinalRelease is called.  The base class will automatically
    // deletes the object.  Add additional cleanup required for your
    // object before calling the base class.

    CCmdTarget::OnFinalRelease();
}


//*****************************************************************************
// Close -  Create a stream in the storage passed in and write state
//          to the stream.
//*****************************************************************************
```

```
BOOL CControl::Close(LPSTORAGE pStg)
{
    USES_CONVERSION;
    LPSTREAM        pStream = NULL ;

// Create a stream to write the control to.
    VERIFY(pStg->CreateStream(T2OLE(key),
                STGM_CREATE | STGM_READWRITE | STGM_SHARE_EXCLUSIVE,
                0, 0, &pStream ) == S_OK);
    ASSERT(pStream != NULL) ;

// Write the version number of our control to the storage stream.
    pStream->Write((DWORD *)&m_Version, sizeof(DWORD), NULL) ;
// Write all data members to the OLE stream
    WriteString(pStream, (LPCTSTR)m_Name) ;
    WriteString(pStream, (LPCTSTR)m_Label) ;
    WriteString(pStream, (LPCTSTR)m_oID) ;
    pStream->Write((short *)&m_xPos,    sizeof(short), NULL) ;
    pStream->Write((short *)&m_yPos,    sizeof(short), NULL) ;
    pStream->Write((short *)&m_Width,   sizeof(short), NULL) ;
    pStream->Write((short *)&m_Height,  sizeof(short), NULL) ;
    pStream->Write((short *)&m_controlType, sizeof(short), NULL) ;

    pStream->Release() ;

    return TRUE ;
}


//*****************************************************************************
// Open -   Read state from the passed in stream
//*****************************************************************************
BOOL CControl::Open(LPSTREAM pStream)
{
    USES_CONVERSION;
    DWORD           version ;
    char            buf[80] ;

// Read the version number
    pStream->Read((DWORD *)&version,    sizeof(DWORD), NULL) ;
    m_Version = version ;

// Read the Control name
    ReadString(pStream, (LPTSTR)&buf) ;
    m_Name = buf ;

// Read the label
    ReadString(pStream, (LPTSTR)&buf) ;
    m_Label = buf ;

// Read the OID
    ReadString(pStream, (LPTSTR)&buf) ;
    m_oID = buf ;
    key = m_oID ;

// Read the xPos
    pStream->Read((short *)&m_xPos,    sizeof(short), NULL) ;

// Read the yPos
    pStream->Read((short *)&m_yPos,    sizeof(short), NULL) ;
```

```
// Read the width
    pStream->Read((short *)&m_Width,  sizeof(short), NULL) ;

// Read the height
    pStream->Read((short *)&m_Height, sizeof(short), NULL) ;

// Read the ControlType
    pStream->Read((short *)&m_controlType, sizeof(short), NULL) ;


    return TRUE ;
}



BEGIN_MESSAGE_MAP(CControl, CCmdTarget)
    //{{AFX_MSG_MAP(CControl)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_DISPATCH_MAP(CControl, CCmdTarget)
    //{{AFX_DISPATCH_MAP(CControl)
    DISP_PROPERTY_NOTIFY(CControl, "XPos", m_xPos, OnXPosChanged, VT_I2)
    DISP_PROPERTY_NOTIFY(CControl, "YPos", m_yPos, OnYPosChanged, VT_I2)
    DISP_PROPERTY_NOTIFY(CControl, "Name", m_Name, OnNameChanged, VT_BSTR)
    DISP_PROPERTY_NOTIFY(CControl, "Label", m_Label, OnLabelChanged, VT_BSTR)
    DISP_PROPERTY_NOTIFY(CControl, "Width", m_Width, OnWidthChanged, VT_I2)
    DISP_PROPERTY_NOTIFY(CControl, "Height", m_Height, OnHeightChanged, VT_I2)
    DISP_PROPERTY_NOTIFY(CControl, "OID", m_oID, OnOIDChanged, VT_BSTR)
    DISP_PROPERTY_NOTIFY(CControl, "ControlType", m_controlType,
    OnControlTypeChanged, VT_I2)
    //}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()

// Note: we add support for IID_IControl to support typesafe binding
//  from VBA.  This IID must match the GUID that is attached to the
//  dispinterface in the .ODL file.

// {96D8F0E0-0A47-11D0-818A-0020AFBACAFF}
//static const IID IID_IControl =
//{ 0x96d8f0e0, 0xa47, 0x11d0, { 0x81, 0x8a, 0x0, 0x20, 0xaf, 0xba, 0xca, 0xff
} };

BEGIN_INTERFACE_MAP(CControl, CCmdTarget)
    INTERFACE_PART(CControl, DIID_IControl, Dispatch)
// 05oct95  npt  Dual interface support
    INTERFACE_PART(CControl, IID_IDualControl, DualControl)
    DUAL_ERRORINFO_PART(CControl)
// End Dual interface support
END_INTERFACE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CControl message handlers

void CControl::OnXPosChanged()
{
    // TODO: Add notification handler code

}
```

```
void CControl::OnYPosChanged()
{
    // TODO: Add notification handler code

}

void CControl::OnNameChanged()
{
    // TODO: Add notification handler code

}

void CControl::OnLabelChanged()
{
    // TODO: Add notification handler code

}

void CControl::OnWidthChanged()
{
    // TODO: Add notification handler code

}

void CControl::OnHeightChanged()
{
    // TODO: Add notification handler code

}


void CControl::OnOIDChanged()
{
    // TODO: Add notification handler code

}

void CControl::SetOID()
{
    m_oID = key ;
}


void CControl::OnControlTypeChanged()
{


}

/////////////////////////////////////////////////////////////////////////////
// DUAL INTERFACE FUNCTIONS
/////////////////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////////////
// delegate standard IDispatch methods to MFC IDispatch implementation
// Macro defined in MFCDUAL.H
DELEGATE_DUAL_INTERFACE(CControl, DualControl)

// Implement ISupportErrorInfo to indicate we support the
// OLE Automation error handler.
IMPLEMENT_DUAL_ERRORINFO(CControl, IID_IDualControl)
```

```
//*****************************************************************************
// put_XPos -
//*****************************************************************************
STDMETHODIMP CControl::XDualControl::put_XPos(short newXPos)
{
    METHOD_PROLOGUE(CControl, DualControl)
    TRY_DUAL(IID_IDualControl)
        {
        pThis->m_xPos = newXPos ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*****************************************************************************
// get_XPos -
//*****************************************************************************
STDMETHODIMP CControl::XDualControl::get_XPos(short* retval)
{
    METHOD_PROLOGUE(CControl, DualControl)
    TRY_DUAL(IID_IDualControl)
        {
        *retval = pThis->m_xPos ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*****************************************************************************
// put_YPos -
//*****************************************************************************
STDMETHODIMP CControl::XDualControl::put_YPos(short newYPos)
{
    METHOD_PROLOGUE(CControl, DualControl)
    TRY_DUAL(IID_IDualControl)
        {
        pThis->m_yPos = newYPos ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*****************************************************************************
// get_YPos -
//*****************************************************************************
STDMETHODIMP CControl::XDualControl::get_YPos(short* curYPos)
{
    METHOD_PROLOGUE(CControl, DualControl)
    TRY_DUAL(IID_IDualControl)
        {
        *curYPos = pThis->m_yPos ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*****************************************************************************
// put_Name -
//*****************************************************************************
STDMETHODIMP CControl::XDualControl::put_Name(BSTR newName)
{
    METHOD_PROLOGUE(CControl, DualControl)
```

```
    TRY_DUAL(IID_IDualControl)
        {
        pThis->m_Name = newName ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*****************************************************************************
// get_Name -
//*****************************************************************************
STDMETHODIMP CControl::XDualControl::get_Name(BSTR* retval)
{
    METHOD_PROLOGUE(CControl, DualControl)
    TRY_DUAL(IID_IDualControl)
        {
//      *retval = pThis->m_Name.AllocSysString() ;
        pThis->m_Name.SetSysString(retval) ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*****************************************************************************
// put_Label -
//*****************************************************************************
STDMETHODIMP CControl::XDualControl::put_Label(BSTR newLabel)
{
    METHOD_PROLOGUE(CControl, DualControl)
    TRY_DUAL(IID_IDualControl)
        {
        pThis->m_Label = newLabel ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*****************************************************************************
// get_Label -
//*****************************************************************************
STDMETHODIMP CControl::XDualControl::get_Label(BSTR* retval)
{
    METHOD_PROLOGUE(CControl, DualControl)
    TRY_DUAL(IID_IDualControl)
        {
//      *retval = pThis->m_Label.AllocSysString() ;
        pThis->m_Label.SetSysString(retval) ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*****************************************************************************
// put_Width -
//*****************************************************************************
STDMETHODIMP CControl::XDualControl::put_Width(short newWidth)
{
    METHOD_PROLOGUE(CControl, DualControl)
    TRY_DUAL(IID_IDualControl)
        {
        pThis->m_Width = newWidth ;
        return NOERROR;
```

```
        }
    CATCH_ALL_DUAL
}

//*****************************************************************************
// get_Width -
//*****************************************************************************
STDMETHODIMP CControl::XDualControl::get_Width(short* curWidth)
{
    METHOD_PROLOGUE(CControl, DualControl)
    TRY_DUAL(IID_IDualControl)
        {
        *curWidth = pThis->m_Width ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*****************************************************************************
// put_Height -
//*****************************************************************************
STDMETHODIMP CControl::XDualControl::put_Height(short newHeight)
{
    METHOD_PROLOGUE(CControl, DualControl)
    TRY_DUAL(IID_IDualControl)
        {
        pThis->m_Height = newHeight ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*****************************************************************************
// get_Height -
//*****************************************************************************
STDMETHODIMP CControl::XDualControl::get_Height(short* curHeight)
{
    METHOD_PROLOGUE(CControl, DualControl)
    TRY_DUAL(IID_IDualControl)
        {
        *curHeight = pThis->m_Height ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*****************************************************************************
// put_OID -
//*****************************************************************************
STDMETHODIMP CControl::XDualControl::put_OID(BSTR newOID)
{
    METHOD_PROLOGUE(CControl, DualControl)
    TRY_DUAL(IID_IDualControl)
        {
        pThis->m_oID = newOID ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*****************************************************************************
// get_OID -
```

```
//*****************************************************************************
STDMETHODIMP CControl::XDualControl::get_OID(BSTR* retval)
{
    METHOD_PROLOGUE(CControl, DualControl)
    TRY_DUAL(IID_IDualControl)
        {
//      *retval = pThis->m_oID.AllocSysString() ;
        pThis->m_oID.SetSysString(retval) ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*****************************************************************************
// put_Type -
//*****************************************************************************
STDMETHODIMP CControl::XDualControl::put_Type(short newType)
{
    METHOD_PROLOGUE(CControl, DualControl)
    TRY_DUAL(IID_IDualControl)
        {
        pThis->m_controlType = newType ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}

//*****************************************************************************
// get_Type -
//*****************************************************************************
STDMETHODIMP CControl::XDualControl::get_Type(short* curType)
{
    METHOD_PROLOGUE(CControl, DualControl)
    TRY_DUAL(IID_IDualControl)
        {
        *curType = pThis->m_controlType ;
        return NOERROR;
        }
    CATCH_ALL_DUAL
}
```

```
========================================================================
       MICROSOFT FOUNDATION CLASS LIBRARY : lit5
========================================================================


AppWizard has created this lit5 DLL for you.  This DLL not only
demonstrates the basics of using the Microsoft Foundation classes but
is also a starting point for writing your DLL.

This file contains a summary of what you will find in each of the files that
make up your lit5 DLL.

lit5.h
    This is the main header file for the DLL.  It declares the
    CLit5App class.

lit5.cpp
    This is the main DLL source file.  It contains the class CLit5App.
    It also contains the OLE entry points required of inproc servers.

lit5.odl
    This file contains the Object Description Language source code for the
    type library of your DLL.

lit5.rc
    This is a listing of all of the Microsoft Windows resources that the
    program uses.  It includes the icons, bitmaps, and cursors that are stored
    in the RES subdirectory.  This file can be directly edited in Microsoft
    Developer Studio.

res\lit5.rc2
    This file contains resources that are not edited by Microsoft
    Developer Studio.  You should place all resources not
    editable by the resource editor in this file.

lit5.odl
    This file contains the Object Description Language source code for the
    type library of your application.

lit5.def
    This file contains information about the DLL that must be
    provided to run with Microsoft Windows.  It defines parameters
    such as the name and description of the DLL.  It also exports
    functions from the DLL.

lit5.clw
    This file contains information used by ClassWizard to edit existing
    classes or add new classes.  ClassWizard also uses this file to store
    information needed to create and edit message maps and dialog data
    maps and to create prototype member functions.

/////////////////////////////////////////////////////////////////////////////
Other standard files:

StdAfx.h, StdAfx.cpp
    These files are used to build a precompiled header (PCH) file
    named lit5.pch and a precompiled types file named StdAfx.obj.

Resource.h
    This is the standard header file, which defines new resource IDs.
    Microsoft Developer Studio reads and updates this file.

/////////////////////////////////////////////////////////////////////////////
Other notes:
```

AppWizard uses "TODO:" to indicate parts of the source code you
should add to or customize.

//

```
// Machine generated IDispatch wrapper class(es) created with ClassWizard
/////////////////////////////////////////////////////////////////////////////
// IProperty wrapper class

class IProperty : public COleDispatchDriver
{
public:
    IProperty() {}      // Calls COleDispatchDriver default constructor
    IProperty(LPDISPATCH pDispatch) : COleDispatchDriver(pDispatch) {}
    IProperty(const IProperty& dispatchSrc) : COleDispatchDriver(dispatchSrc)
    {}

// Attributes
public:
    CString GetName();
    void SetName(LPCTSTR);
    long GetKind();
    void SetKind(long);
    BOOL GetIsDirty();
    void SetIsDirty(BOOL);
    VARIANT GetValue();
    void SetValue(const VARIANT&);
    LPDISPATCH GetStorageInformation();
    void SetStorageInformation(LPDISPATCH);
    long GetPid();
    void SetPid(long);

// Operations
public:
};
/////////////////////////////////////////////////////////////////////////////
// IProperties wrapper class

class IProperties : public COleDispatchDriver
{
public:
    IProperties() {}        // Calls COleDispatchDriver default constructor
    IProperties(LPDISPATCH pDispatch) : COleDispatchDriver(pDispatch) {}
    IProperties(const IProperties& dispatchSrc) :
    COleDispatchDriver(dispatchSrc) {}

// Attributes
public:
    long GetCount();
    void SetCount(long);
    LPUNKNOWN Get_NewEnum();
    void Set_NewEnum(LPUNKNOWN);

// Operations
public:
    LPDISPATCH Add(BSTR* Name, long* PID, long* Kind, VARIANT* Value);
    LPDISPATCH Item(const VARIANT& Index);
    void Remove(const VARIANT& Index);
};
/////////////////////////////////////////////////////////////////////////////
// ILit wrapper class

class ILit : public COleDispatchDriver
{
public:
    ILit() {}       // Calls COleDispatchDriver default constructor
    ILit(LPDISPATCH pDispatch) : COleDispatchDriver(pDispatch) {}
    ILit(const ILit& dispatchSrc) : COleDispatchDriver(dispatchSrc) {}
```

```
// Attributes
public:
    LPDISPATCH GetPropertysets();
    void SetPropertysets(LPDISPATCH);
    LPDISPATCH Get_PropertySets();
    void Set_PropertySets(LPDISPATCH);

// Operations
public:
    SCODE Open(BSTR* LitFileName);
    SCODE Save();
    SCODE SaveAs(BSTR* LitFileName);
    SCODE Close();
};
/////////////////////////////////////////////////////////////////////////////
// IPropertySet wrapper class

class IPropertySet : public COleDispatchDriver
{
public:
    IPropertySet() {}        // Calls COleDispatchDriver default constructor
    IPropertySet(LPDISPATCH pDispatch) : COleDispatchDriver(pDispatch) {}
    IPropertySet(const IPropertySet& dispatchSrc) :
    COleDispatchDriver(dispatchSrc) {}

// Attributes
public:
    CString GetName();
    void SetName(LPCTSTR);
    LPDISPATCH GetProperties();
    void SetProperties(LPDISPATCH);
    LPDISPATCH Get_Properties();
    void Set_Properties(LPDISPATCH);
    long GetStoredAs();
    void SetStoredAs(long);
    CString GetClsid();
    void SetClsid(LPCTSTR);
    CString GetFmtid();
    void SetFmtid(LPCTSTR);
    CString GetForeignPathname();
    void SetForeignPathname(LPCTSTR);
    BOOL GetIsDirty();
    void SetIsDirty(BOOL);

// Operations
public:
};
/////////////////////////////////////////////////////////////////////////////
// IPropertySets wrapper class

class IPropertySets : public COleDispatchDriver
{
public:
    IPropertySets() {}      // Calls COleDispatchDriver default constructor
    IPropertySets(LPDISPATCH pDispatch) : COleDispatchDriver(pDispatch) {}
    IPropertySets(const IPropertySets& dispatchSrc) :
    COleDispatchDriver(dispatchSrc) {}

// Attributes
public:
    long GetCount();
    void SetCount(long);
    LPUNKNOWN Get_NewEnum();
    void Set_NewEnum(LPUNKNOWN);
```

```
// Operations
public:
    LPDISPATCH Add(BSTR* Name, BSTR* CLSID, BSTR* FMTID, long* StoredAs);
    LPDISPATCH Item(const VARIANT& Index);
    void Remove(const VARIANT& Index);
};
```

```
// Machine generated IDispatch wrapper class(es) created with ClassWizard

#include "stdafx.h"
#include "ylit5.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


/////////////////////////////////////////////////////////////////////////////
// IProperty properties

CString IProperty::GetName()
{
    CString result;
    GetProperty(0x1, VT_BSTR, (void*)&result);
    return result;
}

void IProperty::SetName(LPCTSTR propVal)
{
    SetProperty(0x1, VT_BSTR, propVal);
}

long IProperty::GetKind()
{
    long result;
    GetProperty(0x4, VT_I4, (void*)&result);
    return result;
}

void IProperty::SetKind(long propVal)
{
    SetProperty(0x4, VT_I4, propVal);
}

BOOL IProperty::GetIsDirty()
{
    BOOL result;
    GetProperty(0x3, VT_BOOL, (void*)&result);
    return result;
}

void IProperty::SetIsDirty(BOOL propVal)
{
    SetProperty(0x3, VT_BOOL, propVal);
}

VARIANT IProperty::GetValue()
{
    VARIANT result;
    GetProperty(0x2, VT_VARIANT, (void*)&result);
    return result;
}

void IProperty::SetValue(const VARIANT& propVal)
{
    SetProperty(0x2, VT_VARIANT, &propVal);
}
```

```
LPDISPATCH IProperty::GetStorageInformation()
{
    LPDISPATCH result;
    GetProperty(0x5, VT_DISPATCH, (void*)&result);
    return result;
}

void IProperty::SetStorageInformation(LPDISPATCH propVal)
{
    SetProperty(0x5, VT_DISPATCH, propVal);
}

long IProperty::GetPid()
{
    long result;
    GetProperty(0x6, VT_I4, (void*)&result);
    return result;
}

void IProperty::SetPid(long propVal)
{
    SetProperty(0x6, VT_I4, propVal);
}

//////////////////////////////////////////////////////////////////////////
// IProperty operations


//////////////////////////////////////////////////////////////////////////
// IProperties properties

long IProperties::GetCount()
{
    long result;
    GetProperty(0x1, VT_I4, (void*)&result);
    return result;
}

void IProperties::SetCount(long propVal)
{
    SetProperty(0x1, VT_I4, propVal);
}

LPUNKNOWN IProperties::Get_NewEnum()
{
    LPUNKNOWN result;
    GetProperty(0xfffffffc, VT_UNKNOWN, (void*)&result);
    return result;
}

void IProperties::Set_NewEnum(LPUNKNOWN propVal)
{
    SetProperty(0xfffffffc, VT_UNKNOWN, propVal);
}

//////////////////////////////////////////////////////////////////////////
// IProperties operations

LPDISPATCH IProperties::Add(BSTR* Name, long* PID, long* Kind, VARIANT* Value)
{
    LPDISPATCH result;
    static BYTE parms[] =
        VTS_PBSTR VTS_PI4 VTS_PI4 VTS_PVARIANT;
    InvokeHelper(0x3, DISPATCH_METHOD, VT_DISPATCH, (void*)&result, parms,
```

```
        Name, PID, Kind, Value);
    return result;
}

LPDISPATCH IProperties::Item(const VARIANT& Index)
{
    LPDISPATCH result;
    static BYTE parms[] =
        VTS_VARIANT;
    InvokeHelper(0x0, DISPATCH_METHOD, VT_DISPATCH, (void*)&result, parms,
        &Index);
    return result;
}

void IProperties::Remove(const VARIANT& Index)
{
    static BYTE parms[] =
        VTS_VARIANT;
    InvokeHelper(0x5, DISPATCH_METHOD, VT_EMPTY, NULL, parms,
         &Index);
}


////////////////////////////////////////////////////////////////////////////
// ILit properties

LPDISPATCH ILit::GetPropertysets()
{
    LPDISPATCH result;
    GetProperty(0x1, VT_DISPATCH, (void*)&result);
    return result;
}

void ILit::SetPropertysets(LPDISPATCH propVal)
{
    SetProperty(0x1, VT_DISPATCH, propVal);
}

LPDISPATCH ILit::Get_PropertySets()
{
    LPDISPATCH result;
    GetProperty(0x0, VT_DISPATCH, (void*)&result);
    return result;
}

void ILit::Set_PropertySets(LPDISPATCH propVal)
{
    SetProperty(0x0, VT_DISPATCH, propVal);
}

////////////////////////////////////////////////////////////////////////////
// ILit operations

SCODE ILit::Open(BSTR* LitFileName)
{
    SCODE result;
    static BYTE parms[] =
        VTS_PBSTR;
    InvokeHelper(0x2, DISPATCH_METHOD, VT_ERROR, (void*)&result, parms,
        LitFileName);
    return result;
}

SCODE ILit::Save()
```

```
{
    SCODE result;
    InvokeHelper(0x3, DISPATCH_METHOD, VT_ERROR, (void*)&result, NULL);
    return result;
}

SCODE ILit::SaveAs(BSTR* LitFileName)
{
    SCODE result;
    static BYTE parms[] =
        VTS_PBSTR;
    InvokeHelper(0x4, DISPATCH_METHOD, VT_ERROR, (void*)&result, parms,
        LitFileName);
    return result;
}

SCODE ILit::Close()
{
    SCODE result;
    InvokeHelper(0x5, DISPATCH_METHOD, VT_ERROR, (void*)&result, NULL);
    return result;
}


/////////////////////////////////////////////////////////////////////////////
// IPropertySet properties

CString IPropertySet::GetName()
{
    CString result;
    GetProperty(0x5, VT_BSTR, (void*)&result);
    return result;
}

void IPropertySet::SetName(LPCTSTR propVal)
{
    SetProperty(0x5, VT_BSTR, propVal);
}

LPDISPATCH IPropertySet::GetProperties()
{
    LPDISPATCH result;
    GetProperty(0x6, VT_DISPATCH, (void*)&result);
    return result;
}

void IPropertySet::SetProperties(LPDISPATCH propVal)
{
    SetProperty(0x6, VT_DISPATCH, propVal);
}

LPDISPATCH IPropertySet::Get_Properties()
{
    LPDISPATCH result;
    GetProperty(0x0, VT_DISPATCH, (void*)&result);
    return result;
}

void IPropertySet::Set_Properties(LPDISPATCH propVal)
{
    SetProperty(0x0, VT_DISPATCH, propVal);
}

long IPropertySet::GetStoredAs()
```

```
{
    long result;
    GetProperty(0x7, VT_I4, (void*)&result);
    return result;
}

void IPropertySet::SetStoredAs(long propVal)
{
    SetProperty(0x7, VT_I4, propVal);
}

CString IPropertySet::GetClsid()
{
    CString result;
    GetProperty(0x1, VT_BSTR, (void*)&result);
    return result;
}

void IPropertySet::SetClsid(LPCTSTR propVal)
{
    SetProperty(0x1, VT_BSTR, propVal);
}

CString IPropertySet::GetFmtid()
{
    CString result;
    GetProperty(0x2, VT_BSTR, (void*)&result);
    return result;
}

void IPropertySet::SetFmtid(LPCTSTR propVal)
{
    SetProperty(0x2, VT_BSTR, propVal);
}

CString IPropertySet::GetForeignPathname()
{
    CString result;
    GetProperty(0x3, VT_BSTR, (void*)&result);
    return result;
}

void IPropertySet::SetForeignPathname(LPCTSTR propVal)
{
    SetProperty(0x3, VT_BSTR, propVal);
}

BOOL IPropertySet::GetIsDirty()
{
    BOOL result;
    GetProperty(0x4, VT_BOOL, (void*)&result);
    return result;
}

void IPropertySet::SetIsDirty(BOOL propVal)
{
    SetProperty(0x4, VT_BOOL, propVal);
}

////////////////////////////////////////////////////////////////////////////
// IPropertySet operations


////////////////////////////////////////////////////////////////////////////
```

```
// IPropertySets properties

long IPropertySets::GetCount()
{
    long result;
    GetProperty(0x1, VT_I4, (void*)&result);
    return result;
}

void IPropertySets::SetCount(long propVal)
{
    SetProperty(0x1, VT_I4, propVal);
}

LPUNKNOWN IPropertySets::Get_NewEnum()
{
    LPUNKNOWN result;
    GetProperty(0x2, VT_UNKNOWN, (void*)&result);
    return result;
}

void IPropertySets::Set_NewEnum(LPUNKNOWN propVal)
{
    SetProperty(0x2, VT_UNKNOWN, propVal);
}

////////////////////////////////////////////////////////////////////////////
// IPropertySets operations

LPDISPATCH IPropertySets::Add(BSTR* Name, BSTR* CLSID, BSTR* FMTID, long*
StoredAs)
{
    LPDISPATCH result;
    static BYTE parms[] =
        VTS_PBSTR VTS_PBSTR VTS_PBSTR VTS_PI4;
    InvokeHelper(0x3, DISPATCH_METHOD, VT_DISPATCH, (void*)&result, parms,
        Name, CLSID, FMTID, StoredAs);
    return result;
}

LPDISPATCH IPropertySets::Item(const VARIANT& Index)
{
    LPDISPATCH result;
    static BYTE parms[] =
        VTS_VARIANT;
    InvokeHelper(0x4, DISPATCH_METHOD, VT_DISPATCH, (void*)&result, parms,
        &Index);
    return result;
}

void IPropertySets::Remove(const VARIANT& Index)
{
    static BYTE parms[] =
        VTS_VARIANT;
    InvokeHelper(0x5, DISPATCH_METHOD, VT_EMPTY, NULL, parms,
         &Index);
}
```

```
// VariantListT.h

#ifndef _VARIANTLISTT_H_
#define _VARIANTLISTT_H_

#include <afxtempl.h>

typedef CList<LPVARIANT, LPVARIANT&>  VARIANT_List_t;


#endif
```

```
#ifndef _UNICODECONV_H_
#define _UNICODECONV_H_


/*
 * AnsiToUnicode converts the ANSI string pszA to a Unicode string
 * and returns the Unicode string through ppszW. Space for the
 * the converted string is allocated by AnsiToUnicode.
 */
HRESULT AnsiToUnicode(LPCSTR pszA, LPOLESTR* ppszW);

/*
 * UnicodeToAnsi converts the Unicode string pszW to an ANSI string
 * and returns the ANSI string through ppszA. Space for the
 * the converted string is allocated by UnicodeToAnsi.
 */
HRESULT  UnicodeToAnsi(LPCOLESTR pszW, LPSTR* ppszA);

#endif
```

```
#include "stdafx.h"
#include "UnicodeConv.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/*
 * AnsiToUnicode converts the ANSI string pszA to a Unicode string
 * and returns the Unicode string through ppszW. Space for the
 * the converted string is allocated by AnsiToUnicode.
 */
HRESULT AnsiToUnicode(LPCSTR pszA, LPOLESTR* ppszW)
{
    ULONG cCharacters;
    DWORD dwError;

    // If input is null then just return the same.
    if (NULL == pszA)
    {
        *ppszW = NULL;
        return NOERROR;
    }

    // Determine number of wide characters to be allocated for the
    // Unicode string.
    cCharacters =  strlen(pszA)+1;

    // Use of the OLE allocator is required if the resultant Unicode
    // string will be passed to another COM component and if that
    // component will free it. Otherwise you can use your own allocator.
    *ppszW = (LPOLESTR) CoTaskMemAlloc(cCharacters*2);
    if (NULL == *ppszW)
        return E_OUTOFMEMORY;

    // Covert to Unicode.
    if (0 == MultiByteToWideChar(CP_ACP, 0, pszA, cCharacters,
                  *ppszW, cCharacters))
    {
        dwError = GetLastError();
        CoTaskMemFree(*ppszW);
        *ppszW = NULL;
        return HRESULT_FROM_WIN32(dwError);
    }

    return NOERROR;
}

/*
 * UnicodeToAnsi converts the Unicode string pszW to an ANSI string
 * and returns the ANSI string through ppszA. Space for the
 * the converted string is allocated by UnicodeToAnsi.
 */
HRESULT  UnicodeToAnsi(LPCOLESTR pszW, LPSTR* ppszA)
{
    ULONG cbAnsi, cCharacters;
    DWORD dwError;

    // If input is null then just return the same.
    if (pszW == NULL)
    {
        *ppszA = NULL;
```

```
        return NOERROR;
    }

    cCharacters = wcslen(pszW)+1;
    // Determine number of bytes to be allocated for ANSI string. An
    // ANSI string can have at most 2 bytes per character (for Double
    // Byte Character Strings.)
    cbAnsi = cCharacters*2;

    // Use of the OLE allocator is not required because the resultant
    // ANSI  string will never be passed to another COM component. You
    // can use your own allocator.
    *ppszA = (LPSTR) CoTaskMemAlloc(cbAnsi);
    if (NULL == *ppszA)
        return E_OUTOFMEMORY;

    // Convert to ANSI.
    if (0 == WideCharToMultiByte(CP_ACP, 0, pszW, cCharacters, *ppszA,
                  cbAnsi, NULL, NULL))
    {
        dwError = GetLastError();
        CoTaskMemFree(*ppszA);
        *ppszA = NULL;
        return HRESULT_FROM_WIN32(dwError);
    }

    return NOERROR;
}
```

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#define VC_EXTRALEAN        // Exclude rarely-used stuff from Windows headers

#include <afxole.h>
#include <afxpriv.h>

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions

#ifndef _AFX_NO_OLE_SUPPORT
#include <afxole.h>         // MFC OLE classes
#include <afxodlgs.h>       // MFC OLE dialog classes
#include <afxdisp.h>        // MFC OLE automation classes
#endif // _AFX_NO_OLE_SUPPORT


#ifndef _AFX_NO_DB_SUPPORT
#include <afxdb.h>          // MFC ODBC database classes
#endif // _AFX_NO_DB_SUPPORT

#ifndef _AFX_NO_DAO_SUPPORT
#include <afxdao.h>         // MFC DAO database classes
#endif // _AFX_NO_DAO_SUPPORT

#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>         // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT
```

```
// stdafx.cpp : source file that includes just the standard includes
//  lit5.pch will be the pre-compiled header
//  stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by LIT5.RC
//

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS

#define _APS_NEXT_RESOURCE_VALUE        129
#define _APS_NEXT_COMMAND_VALUE         32771
#define _APS_NEXT_CONTROL_VALUE         1000
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

```
// PropertySets.h : header file
//
#include "VariantListT.h"

/////////////////////////////////////////////////////////////////////////////
// CPropertySets command target

class CPropertySets : public CCmdTarget
{
    DECLARE_DYNCREATE(CPropertySets)

    CPropertySets();           // protected constructor used by dynamic
    creation

// Attributes
public:
  VARIANT_List_t   *m_PropertySet_List;

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CPropertySets)
    public:
    virtual void OnFinalRelease();
    //}}AFX_VIRTUAL

// Implementation
protected:
    virtual ~CPropertySets();

    // Generated message map functions
    //{{AFX_MSG(CPropertySets)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG

    DECLARE_MESSAGE_MAP()
    // Generated OLE dispatch map functions
    //{{AFX_DISPATCH(CPropertySets)
    afx_msg LPUNKNOWN _NewEnum();
    afx_msg long GetCount();
    afx_msg void SetCount(long nNewValue);
    afx_msg void SetNewEnum(LPUNKNOWN newValue);
    afx_msg LPDISPATCH Add(BSTR FAR* Name, BSTR FAR* CLSID, BSTR FAR* FMTID,
    long* StoredAs);
    afx_msg LPDISPATCH Item(const VARIANT FAR& Index);
    afx_msg void Remove(const VARIANT FAR& Index);
    //}}AFX_DISPATCH


    DECLARE_DISPATCH_MAP()
    DECLARE_INTERFACE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

```
// PropertySets.cpp : implementation file
//

#include "stdafx.h"
#include "lit5.h"
#include "PropertySets.h"
#include "PropertySet.h"
#include "EnumVARIANT.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CPropertySets

IMPLEMENT_DYNCREATE(CPropertySets, CCmdTarget)

CPropertySets::CPropertySets()
{
    EnableAutomation();
  m_PropertySet_List = new VARIANT_List_t;
}

CPropertySets::~CPropertySets()
{
}


void CPropertySets::OnFinalRelease()
{
    // When the last reference for an automation object is released
    // OnFinalRelease is called.  The base class will automatically
    // deletes the object.  Add additional cleanup required for your
    // object before calling the base class.

    CCmdTarget::OnFinalRelease();
}


BEGIN_MESSAGE_MAP(CPropertySets, CCmdTarget)
    //{{AFX_MSG_MAP(CPropertySets)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_DISPATCH_MAP(CPropertySets, CCmdTarget)
    //{{AFX_DISPATCH_MAP(CPropertySets)
    DISP_PROPERTY_EX(CPropertySets, "Count", GetCount, SetCount, VT_I4)
    DISP_PROPERTY_EX(CPropertySets, "_NewEnum", _NewEnum, SetNewEnum,
    VT_UNKNOWN)
    DISP_FUNCTION(CPropertySets, "Add", Add, VT_DISPATCH, VTS_PBSTR VTS_PBSTR
    VTS_PBSTR VTS_PI4)
    DISP_FUNCTION(CPropertySets, "Item", Item, VT_DISPATCH, VTS_VARIANT)
    DISP_FUNCTION(CPropertySets, "Remove", Remove, VT_EMPTY, VTS_VARIANT)
    //}}AFX_DISPATCH_MAP
  DISP_PROPERTY_EX_ID(CPropertySets, "_NewEnum", DISPID_NEWENUM,
                      _NewEnum, SetNotSupported, VT_UNKNOWN)
  DISP_DEFVALUE(CPropertySets, "Item")
END_DISPATCH_MAP()

// Note: we add support for IID_IPropertySets to support typesafe binding
```

```
//  from VBA.  This IID must match the GUID that is attached to the
//  dispinterface in the .ODL file.

// {1FA35CA3-0AE6-11D0-BE99-0020AFD208B9}
static const IID IID_IPropertySets =
{ 0x1fa35ca3, 0xae6, 0x11d0, { 0xbe, 0x99, 0x0, 0x20, 0xaf, 0xd2, 0x8, 0xb9 }
};

BEGIN_INTERFACE_MAP(CPropertySets, CCmdTarget)
    INTERFACE_PART(CPropertySets, IID_IPropertySets, Dispatch)
END_INTERFACE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPropertySets message handlers

long CPropertySets::GetCount()
{
    // TODO: Add your property handler here

    return m_PropertySet_List->GetCount();
}

void CPropertySets::SetCount(long nNewValue)
{
    // TODO: Add your property handler here
  ASSERT(0);
}

LPUNKNOWN CPropertySets::_NewEnum()
{
    // TODO: Add your property handler here

    CEnumVARIANT*  pEnumVAR;
    LPUNKNOWN      pIUknown;

    pEnumVAR = new CEnumVARIANT(this->m_PropertySet_List);
    pEnumVAR->ExternalQueryInterface(&IID_IEnumVARIANT,
                                     (void **) &pIUknown);
    pEnumVAR->ExternalRelease(); // balance reference count

    return pIUknown;
}

void CPropertySets::SetNewEnum(LPUNKNOWN newValue)
{
    // TODO: Add your property handler here
   ASSERT(0);
}

LPDISPATCH CPropertySets::Add(BSTR FAR* Name,
          BSTR FAR* CLSID, BSTR FAR* FMTID, long* StoredAs)
{
    // TODO: Add your dispatch handler code here
    // TODO: Add your dispatch handler code here
    CPropertySet  *pPropSetTemp;
    LPDISPATCH     pDispatchTemp;
    LPVARIANT      pTempVariant;

    pPropSetTemp = new CPropertySet(Name, CLSID, FMTID, StoredAs);
      pDispatchTemp = pPropSetTemp->GetIDispatch(TRUE);
    // NOTE:: be sure to delete VARIANT when unloading.
    pTempVariant  = new VARIANT;
    VariantInit(pTempVariant);
    pTempVariant->vt = VT_DISPATCH;
```

```
    pTempVariant->pdispVal = pDispatchTemp;
    m_PropertySet_List->AddTail(pTempVariant);

    return pDispatchTemp;
}

LPDISPATCH CPropertySets::Item(const VARIANT FAR& Index)
{
    // Collections are one-based indexing
     long         ZeroBasedIndex;
     LPVARIANT    pVar;
     LPDISPATCH   result;

     switch (Index.vt) {
       case VT_I2:
                  ZeroBasedIndex = Index.iVal - 1;
                  break;
       case VT_I4:
                  ZeroBasedIndex = Index.lVal - 1;
                  break;
       case (VT_BYREF | VT_I2):
                  ZeroBasedIndex = *(Index.piVal) - 1;
                  break;
       case (VT_BYREF | VT_I4):
                  ZeroBasedIndex = *(Index.plVal) - 1;
                  break;
       default:   return NULL;

     } /* switch */

     // check if out of range
     if ( ZeroBasedIndex < 0  &&
          ZeroBasedIndex >= m_PropertySet_List->GetCount()
        ) {
        return NULL;
     } /* if */

     POSITION pos;
     pos = m_PropertySet_List->FindIndex(ZeroBasedIndex);
     if (NULL == pos) return NULL;
     pVar = m_PropertySet_List->GetAt(pos);
     ASSERT(VT_DISPATCH == pVar->vt);
     result = pVar->pdispVal;
     result->AddRef();
     return result;
}

void CPropertySets::Remove(const VARIANT FAR& Index)
{
    // Collections are one-based indexing
     long         ZeroBasedIndex;
     LPVARIANT    pVar;
     POSITION     pos;

     switch (Index.vt) {
       case VT_I2:
                  ZeroBasedIndex = Index.iVal - 1;
                  break;
       case VT_I4:
                  ZeroBasedIndex = Index.lVal - 1;
                  break;
       case (VT_BYREF | VT_I2):
                  ZeroBasedIndex = *(Index.piVal) - 1;
                  break;
```

```
        case (VT_BYREF | VT_I4):
                    ZeroBasedIndex = *(Index.plVal) - 1;
                    break;
        default:    return;

    } /* switch */

    // check if out of range
    if (  ZeroBasedIndex < 0  &&
          ZeroBasedIndex >= m_PropertySet_List->GetCount()
       ) {
       return;
    } /* if */

    pos = m_PropertySet_List->FindIndex(ZeroBasedIndex);
    pVar = m_PropertySet_List->GetAt(pos);
    ASSERT(VT_DISPATCH == pVar->vt);
    m_PropertySet_List->RemoveAt(pos);
    pVar->pdispVal->Release();

}
```

```
// PropertySet.h : header file
//


/////////////////////////////////////////////////////////////////////////////
// CPropertySet command target

class CPropertySet : public CCmdTarget
{
    DECLARE_DYNCREATE(CPropertySet)

    CPropertySet();   // protected constructor used by dynamic creation
    CPropertySet(BSTR *Name, BSTR *CLSID, BSTR *FMTID,
                long* StoredAs);

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CPropertySet)
    public:
    virtual void OnFinalRelease();
    //}}AFX_VIRTUAL

// Implementation
protected:
    virtual     ~CPropertySet();
  long          m_StoredAs;
  CString       m_Name;
  LPDISPATCH    m_Properties;
    // Generated message map functions
    //{{AFX_MSG(CPropertySet)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG

    DECLARE_MESSAGE_MAP()
    // Generated OLE dispatch map functions
    //{{AFX_DISPATCH(CPropertySet)
    CString m_CLSID;
    afx_msg void OnCLSIDChanged();
    CString m_FMTID;
    afx_msg void OnFMTIDChanged();
    CString m_foreignPathname;
    afx_msg void OnForeignPathnameChanged();
    BOOL m_isDirty;
    afx_msg void OnIsDirtyChanged();
    afx_msg BSTR GetName();
    afx_msg void SetName(LPCTSTR lpszNewValue);
    afx_msg LPDISPATCH GetProperties();
    afx_msg void SetProperties(LPDISPATCH newValue);
    afx_msg long GetStoredAs();
    afx_msg void SetStoredAs(long nNewValue);
    //}}AFX_DISPATCH
    DECLARE_DISPATCH_MAP()
    DECLARE_INTERFACE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

```
// PropertySet.cpp : implementation file
//

#include "stdafx.h"
#include "lit5.h"
#include "PropertySet.h"
#include "Properties.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CPropertySet

IMPLEMENT_DYNCREATE(CPropertySet, CCmdTarget)

CPropertySet::CPropertySet()
{
    EnableAutomation();
}

CPropertySet::CPropertySet(BSTR *Name, BSTR *CLSID, BSTR *FMTID,
                           long* StoredAs)
{
    EnableAutomation();
  CProperties *temp;
  temp = new CProperties;
  m_Properties = temp->GetIDispatch(TRUE);

  m_Name =    *Name;
  m_CLSID =   *CLSID;
  m_FMTID =   *FMTID;
  m_StoredAs ='*StoredAs;
}



CPropertySet::~CPropertySet()
{
}


void CPropertySet::OnFinalRelease()
{
    // When the last reference for an automation object is released
    // OnFinalRelease is called.  The base class will automatically
    // deletes the object.  Add additional cleanup required for your
    // object before calling the base class.

    CCmdTarget::OnFinalRelease();
}


BEGIN_MESSAGE_MAP(CPropertySet, CCmdTarget)
    //{{AFX_MSG_MAP(CPropertySet)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_DISPATCH_MAP(CPropertySet, CCmdTarget)
```

```
    //{{AFX_DISPATCH_MAP(CPropertySet)
    DISP_PROPERTY_NOTIFY(CPropertySet, "CLSID", m_CLSID, OnCLSIDChanged,
    VT_BSTR)
    DISP_PROPERTY_NOTIFY(CPropertySet, "FMTID", m_FMTID, OnFMTIDChanged,
    VT_BSTR)
    DISP_PROPERTY_NOTIFY(CPropertySet, "ForeignPathname", m_foreignPathname,
    OnForeignPathnameChanged, VT_BSTR)
    DISP_PROPERTY_NOTIFY(CPropertySet, "isDirty", m_isDirty, OnIsDirtyChanged,
    VT_BOOL)
    DISP_PROPERTY_EX(CPropertySet, "Name", GetName, SetName, VT_BSTR)
    DISP_PROPERTY_EX(CPropertySet, "Properties", GetProperties, SetProperties,
    VT_DISPATCH)
    DISP_PROPERTY_EX(CPropertySet, "StoredAs", GetStoredAs, SetStoredAs,
    VT_I4)
    DISP_DEFVALUE(CPropertySet, "Properties")
    //}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()

// Note: we add support for IID_IPropertySet to support typesafe binding
//  from VBA.  This IID must match the GUID that is attached to the
//  dispinterface in the .ODL file.

// {1FA35CA0-0AE6-11D0-BE99-0020AFD208B9}
static const IID IID_IPropertySet =
{ 0x1fa35ca0, 0xae6, 0x11d0, { 0xbe, 0x99, 0x0, 0x20, 0xaf, 0xd2, 0x8, 0xb9 }
};

BEGIN_INTERFACE_MAP(CPropertySet, CCmdTarget)
    INTERFACE_PART(CPropertySet, IID_IPropertySet, Dispatch)
END_INTERFACE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPropertySet message handlers

BSTR CPropertySet::GetName()
{
    CString strResult;
    // TODO: Add your property handler here
  strResult = m_Name;
    return strResult.AllocSysString();
}

void CPropertySet::SetName(LPCTSTR lpszNewValue)
{
    // TODO: Add your property handler here
   m_Name = lpszNewValue;
}


LPDISPATCH CPropertySet::GetProperties()
{
    // TODO: Add your property handler here
  m_Properties->AddRef();
    return m_Properties;
}

void CPropertySet::SetProperties(LPDISPATCH newValue)
{
    // TODO: Add your property handler here
  m_Properties = m_Properties;
}

long CPropertySet::GetStoredAs()
{
```

```
    // TODO: Add your property handler here

    return m_StoredAs;
}

void CPropertySet::SetStoredAs(long nNewValue)
{
    // TODO: Add your property handler here
  m_StoredAs = nNewValue;
}

void CPropertySet::OnCLSIDChanged()
{
    // TODO: Add notification handler code

}

void CPropertySet::OnFMTIDChanged()
{
    // TODO: Add notification handler code

}

void CPropertySet::OnForeignPathnameChanged()
{
    // TODO: Add notification handler code

}


void CPropertySet::OnIsDirtyChanged()
{
    // TODO: Add notification handler code

}
```

```
// Property.h : header file
//


/////////////////////////////////////////////////////////////////////////////
// CProperty command target

class CProperty : public CCmdTarget
{
    DECLARE_DYNCREATE(CProperty)

    CProperty();           // protected constructor used by dynamic creation
    CProperty(BSTR *Name, long* PID, long* Kind, VARIANT *value);
// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CProperty)
    public:
    virtual void OnFinalRelease();
    //}}AFX_VIRTUAL

// Implementation
protected:
    virtual      ~CProperty();
  long         m_kind;
  LPDISPATCH   m_StorageInformation;
  long         m_PID;
    // Generated message map functions
    //{{AFX_MSG(CProperty)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG

    DECLARE_MESSAGE_MAP()
    // Generated OLE dispatch map functions
    //{{AFX_DISPATCH(CProperty)
    CString m_name;
    afx_msg void OnNameChanged();
    VARIANT m_value;
    afx_msg void OnValueChanged();
    BOOL m_isDirty;
    afx_msg void OnIsDirtyChanged();
    afx_msg long GetKind();
    afx_msg void SetKind(long nNewValue);
    afx_msg LPDISPATCH GetStorageInformation();
    afx_msg void SetStorageInformation(LPDISPATCH newValue);
    afx_msg long GetPID();
    afx_msg void SetPID(long nNewValue);
    //}}AFX_DISPATCH
    DECLARE_DISPATCH_MAP()
    DECLARE_INTERFACE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

```
// Property.cpp : implementation file
//

#include "stdafx.h"
#include "lit5.h"
#include "Property.h"
#include "UnicodeConv.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CProperty

IMPLEMENT_DYNCREATE(CProperty, CCmdTarget)

CProperty::CProperty()
{
    EnableAutomation();
}

CProperty::CProperty(BSTR *Name, long* PID, long* Kind,
                     VARIANT *value)
{
    EnableAutomation();

  char *tempAnsi;
  // space for tempAnsi is allocated by UnicodeToAnsi
  // NOTE:: be sure to do a CoMemFree on it on unload.
  VERIFY (S_OK == UnicodeToAnsi(*Name, &tempAnsi));
  m_name = tempAnsi;

  m_kind  = *Kind;
  m_PID = *PID;
  VariantInit(&m_value);
  VERIFY(S_OK == VariantCopy(&m_value, value));
}


CProperty::~CProperty()
{
}


void CProperty::OnFinalRelease()
{
    // When the last reference for an automation object is released
    // OnFinalRelease is called.  The base class will automatically
    // deletes the object.  Add additional cleanup required for your
    // object before calling the base class.

    CCmdTarget::OnFinalRelease();
}


BEGIN_MESSAGE_MAP(CProperty, CCmdTarget)
    //{{AFX_MSG_MAP(CProperty)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
BEGIN_DISPATCH_MAP(CProperty, CCmdTarget)
    //{{AFX_DISPATCH_MAP(CProperty)
    DISP_PROPERTY_NOTIFY(CProperty, "Name", m_name, OnNameChanged, VT_BSTR)
    DISP_PROPERTY_NOTIFY(CProperty, "Value", m_value, OnValueChanged,
    VT_VARIANT)
    DISP_PROPERTY_NOTIFY(CProperty, "isDirty", m_isDirty, OnIsDirtyChanged,
    VT_BOOL)
    DISP_PROPERTY_EX(CProperty, "Kind", GetKind, SetKind, VT_I4)
    DISP_PROPERTY_EX(CProperty, "StorageInformation", GetStorageInformation,
    SetStorageInformation, VT_DISPATCH)
    DISP_PROPERTY_EX(CProperty, "PID", GetPID, SetPID, VT_I4)
    //}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()

// Note: we add support for IID_IProperty to support typesafe binding
//  from VBA.  This IID must match the GUID that is attached to the
//  dispinterface in the .ODL file.

// {A10DE2C4-FFE1-11CF-BE99-0020AFD208B9}
static const IID IID_IProperty =
{ 0xa10de2c4, 0xffe1, 0x11cf, { 0xbe, 0x99, 0x0, 0x20, 0xaf, 0xd2, 0x8, 0xb9 }
};

BEGIN_INTERFACE_MAP(CProperty, CCmdTarget)
    INTERFACE_PART(CProperty, IID_IProperty, Dispatch)
END_INTERFACE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CProperty message handlers

void CProperty::OnNameChanged()
{
    // TODO: Add notification handler code

}

long CProperty::GetKind()
{
    // TODO: Add your property handler here

    return m_kind;
}

void CProperty::SetKind(long nNewValue)
{
    // TODO: Add your property handler here
      m_kind = nNewValue;
}

void CProperty::OnValueChanged()
{
    // TODO: Add notification handler code

}

LPDISPATCH CProperty::GetStorageInformation()
{
    // TODO: Add your property handler here
    return m_StorageInformation;
}

void CProperty::SetStorageInformation(LPDISPATCH newValue)
{
    // TODO: Add your property handler here
```

```
  m_StorageInformation = newValue;
}

long CProperty::GetPID()
{
    // TODO: Add your property handler here

    return m_PID;
}

void CProperty::SetPID(long nNewValue)
{
    // TODO: Add your property handler here
  m_PID = nNewValue;

}

void CProperty::OnIsDirtyChanged()
{
    // TODO: Add notification handler code

}
```

```
// Properties.h : header file
//
#ifndef _PROPERTIES_H_
#define _PROPERTIES_H_

#include "VariantListT.h"

/////////////////////////////////////////////////////////////////////////////
// CProperties command target

class CProperties : public CCmdTarget
{
    DECLARE_DYNCREATE(CProperties)

    CProperties();           // protected constructor used by dynamic creation

// Attributes
public:
    VARIANT_List_t   *m_Property_List;

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CProperties)
    public:
    virtual void OnFinalRelease();
    //}}AFX_VIRTUAL

// Implementation
protected:
    virtual ~CProperties();

    // Generated message map functions
    //{{AFX_MSG(CProperties)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG

    DECLARE_MESSAGE_MAP()
    // Generated OLE dispatch map functions
    //{{AFX_DISPATCH(CProperties)
    afx_msg long GetCount();
    afx_msg void SetCount(long nNewValue);
    afx_msg LPUNKNOWN _NewEnum();
    afx_msg void SetNewEnum(LPUNKNOWN newValue);
    afx_msg LPDISPATCH Add(BSTR FAR* Name, long FAR* PID, long FAR* Kind,
    VARIANT FAR* Value);
    afx_msg LPDISPATCH Item(const VARIANT FAR& Index);
    afx_msg void Remove(const VARIANT FAR& Index);
    //}}AFX_DISPATCH
    DECLARE_DISPATCH_MAP()



    DECLARE_INTERFACE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
#endif
```

```
// Properties.cpp : implementation file
//

#include "stdafx.h"
#include "lit5.h"
#include "Properties.h"
#include "Property.h"
#include "EnumVARIANT.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CProperties

IMPLEMENT_DYNCREATE(CProperties, CCmdTarget)

CProperties::CProperties()
{
    EnableAutomation();
  m_Property_List = new VARIANT_List_t;
}

CProperties::~CProperties()
{
}


void CProperties::OnFinalRelease()
{
    // When the last reference for an automation object is released
    // OnFinalRelease is called.  The base class will automatically
    // deletes the object.  Add additional cleanup required for your
    // object before calling the base class.

    CCmdTarget::OnFinalRelease();
}


BEGIN_MESSAGE_MAP(CProperties, CCmdTarget)
    //{{AFX_MSG_MAP(CProperties)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_DISPATCH_MAP(CProperties, CCmdTarget)
    //{{AFX_DISPATCH_MAP(CProperties)
    DISP_PROPERTY_EX(CProperties, "Count", GetCount, SetCount, VT_I4)
    DISP_PROPERTY_EX(CProperties, "_NewEnum", _NewEnum, SetNewEnum,
    VT_UNKNOWN)
    DISP_FUNCTION(CProperties, "Add", Add, VT_DISPATCH, VTS_PBSTR VTS_PI4
    VTS_PI4 VTS_PVARIANT)
    DISP_FUNCTION(CProperties, "Item", Item, VT_DISPATCH, VTS_VARIANT)
    DISP_FUNCTION(CProperties, "Remove", Remove, VT_EMPTY, VTS_VARIANT)
  DISP_DEFVALUE(CProperties, "Item")
    //}}AFX_DISPATCH_MAP
  DISP_PROPERTY_EX_ID(CProperties, "_NewEnum", DISPID_NEWENUM, _NewEnum,
  SetNotSupported, VT_UNKNOWN)
  DISP_DEFVALUE(CProperties, "Item")
END_DISPATCH_MAP()
```

```
// Note: we add support for IID_IProperties to support typesafe binding
//  from VBA.  This IID must match the GUID that is attached to the
//  dispinterface in the .ODL file.

// {A10DE2C7-FFE1-11CF-BE99-0020AFD208B9}
static const IID IID_IProperties =
{ 0xa10de2c7, 0xffe1, 0x11cf, { 0xbe, 0x99, 0x0, 0x20, 0xaf, 0xd2, 0x8, 0xb9 }
};

BEGIN_INTERFACE_MAP(CProperties, CCmdTarget)
    INTERFACE_PART(CProperties, IID_IProperties, Dispatch)
END_INTERFACE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CProperties message handlers

LPDISPATCH CProperties::Add(BSTR FAR*    Name,
                            long FAR*    PID,
                            long FAR*    Kind,
                            VARIANT FAR* Value)
{
    // TODO: Add your dispatch handler code here
    CProperty   *pPropTemp;
    LPDISPATCH  pDispatchTemp;
    LPVARIANT   pTempVariant;

    pPropTemp = new CProperty(Name, PID, Kind, Value);
      pDispatchTemp = pPropTemp->GetIDispatch(TRUE);
    // NOTE:: be sure to delete VARIANT when unloading.
    pTempVariant  = new VARIANT;
    VariantInit(pTempVariant);
    pTempVariant->vt = VT_DISPATCH;
    pTempVariant->pdispVal = pDispatchTemp;
    m_Property_List->AddTail(pTempVariant);

    return pDispatchTemp;
}


long CProperties::GetCount()
{
    // TODO: Add your property handler here
    return m_Property_List->GetCount();

}

void CProperties::SetCount(long nNewValue)
{
    // TODO: Add your property handler here
    ASSERT(0);
}

LPDISPATCH CProperties::Item(const VARIANT FAR& Index)
{
    // Collections are one-based indexing
     long          ZeroBasedIndex;
     LPVARIANT     pVar;
     LPDISPATCH    result;

     switch (Index.vt) {
       case VT_I2:
                   ZeroBasedIndex = Index.iVal - 1;
                   break;
       case VT_I4:
```

```
                ZeroBasedIndex = Index.lVal - 1;
                break;
        case (VT_BYREF | VT_I2):
                ZeroBasedIndex = *(Index.piVal) - 1;
                break;
        case (VT_BYREF | VT_I4):
                ZeroBasedIndex = *(Index.plVal) - 1;
                break;
        default:    return NULL;

      } /* switch */

      // check if out of range
      if (  ZeroBasedIndex < 0  &&
            ZeroBasedIndex >= m_Property_List->GetCount()
         ) {
         return NULL;
      } /* if */

      POSITION pos;
      pos = m_Property_List->FindIndex(ZeroBasedIndex);
      if (NULL == pos) return NULL;
      pVar = m_Property_List->GetAt(pos);
      ASSERT(VT_DISPATCH == pVar->vt);
      result = pVar->pdispVal;
      result->AddRef();
      return result;
}

void CProperties::Remove(const VARIANT FAR& Index)
{
    // Collections are one-based indexing
      long          ZeroBasedIndex;
      LPVARIANT     pVar;
      POSITION      pos;

      switch (Index.vt) {
        case VT_I2:
                ZeroBasedIndex = Index.iVal - 1;
                break;
        case VT_I4:
                ZeroBasedIndex = Index.lVal - 1;
                break;
        case (VT_BYREF | VT_I2):
                ZeroBasedIndex = *(Index.piVal) - 1;
                break;
        case (VT_BYREF | VT_I4):
                ZeroBasedIndex = *(Index.plVal) - 1;
                break;
        default:    return;

      } /* switch */

      // check if out of range
      if (  ZeroBasedIndex < 0  &&
            ZeroBasedIndex >= m_Property_List->GetCount()
         ) {
         return;
      } /* if */

      pos = m_Property_List->FindIndex(ZeroBasedIndex);
      pVar = m_Property_List->GetAt(pos);
      ASSERT(VT_DISPATCH == pVar->vt);
      m_Property_List->RemoveAt(pos);
```

APPENDIX B (Object-Based Parameters)
to
"SYSTEM and METHODS for MANAGING DIGITAL CREATIVE WORKS"
by John S. Erickson

Filed on October 11, 1996

1. Requirements

1.1. General System Requirements

1.1.1. The Needs Addressed by LicensIt

The fundamental goal of the process of rights management is to protect the rights of intellectual property owners while promoting open and free sharing of information. In practical terms, this means that the benefits of creating works, tangible and otherwise, should accrue to their authors, and the benefits of acquiring and disseminating works to their publishers, without reducing the quality and availability of content.

While remunerating and crediting authors does indeed spur further creation and promote both quality and quantity of content, cumbersome methods of doing so inhibit open and free availability. This is especially true in the area of digital multimedia. There is a great need for efficient tools to acquire, publish, distribute, disseminate, and consume multimedia works — tools that strengthen ownership and attribution and enhance the relationships between owner/creators and developer/publishers.

From the point of view of those who deal with multimedia rights management, the issues fall into four categories: identification, access, business, and systems/technology.

- Identification: Potential licensees must be able to identify a work's owner, must be able to identify creative contributors to evaluate a work, must be able to ascertain authenticity of the work and its identifying information. Authors and publishers are concerned with the permanence of identification through derivative uses.

- Access: All those involved wish to maximize the ability to locate and evaluate work.

- Business: All those involved wish to retain maximum flexibility of fees and terms, from simple and automatic transactions to complex and individual negotiations. Works should engender referral for additional work, modifications or non-standard renderings.

- Systems/technology: Any system to address these issues must support without restriction the choice and use of current and future formats, systems, and tools. It should support and comply with technological and industrial standards.

The current methodology of rights management, based on hard copy technology, involves attaching attribution and notification to works, such as copyright notices, by lines and credits. This process is certainly not flawless. Such notices become dated, and can usually be removed and/or ignored. In the event that they fail, it can be difficult to determine when or how a violation occurred.

Digital media exacerbate these problems. Digital works may be exact (or perhaps undetectably altered) copies, or inauthentic or unauthorized replications. With current technology, such copies are easy to make. Furthermore, the network rapidly dissociates the author from the work. So, the current methodology's insufficiencies are magnified by the digital environment.

It is the purpose of LicensIt to address these problems and apply additional computing capabilities to rights management. LicensIt is meant to lower the cost and enhance the ease and effectiveness of the following fundamental operations:

- Creators of multimedia works attaching copyright notices and other attributes and properties to their work.

- Creators of derivative work locating source works.
- Creators of derivative work obtaining releases and permissions to incorporate another work or part of another work in their work.
- Consumers of a work ascertaining the validity and authenticity of the work.
- Consumers of a work determining the rights and restrictions concerning its use.
- Consumers of a work, either for end use or in composing derivative work, determining the source and other attributes of the work.

LicensIt also adds the following capabilities to the process of rights management:

- To invoke additional services, such as email from consumers to authors or rights owners.
- To provide digital security/authentication of transactions and works.
- To generally automate the processes of rights management, such as acquisition, administration, and authorization.

1.1.2. How LicensIt Addresses the Needs

LicensIt addresses the needs of rights management by applying object technology to digital media. From the LicensIt perspective, a work is an object consisting of its content and other attributes, and whose methods comprise the services and information available to both the LicensIt UI and the LicensIt API. The attributes and content of a LicensIt object are distributed between the local system, where it is viewed or manipulated, and a LicensIt Registry service to which it refers across the Internet. The Registry contains attributes that, for various reasons (such as volatility, security, or efficiency), cannot travel to the local system. Finally, a Repository system provides file images (persistence data) of LicensIt objects as well as resources and data referred to by LicensIt objects but not held in attributes at the Registry.

The LicensIt Desktop UI is accessed via a set of Property Pages (tabbed dialog boxes) available from the LicensIt object's representation wherever it appears. For example, where an icon for a LicensIt object appears in the shell or a screen rendering appears in a browser, a context click (right mouse click in Windows 95) brings up property pages that show information and provide access to features such as email and authentication.

Creators bind content and attributes into a LicensIt object and register new objects with the LicensIt Creator's Tool Box, which facilitates the flexible design of an object's Property Pages and selection of its LicensIt features. When LicensIt objects are being composed or included into derivative work, the Tool Box also automates the organization and maintenance of the derivative's heritage.

As mentioned above, many LicensIt objects are registered on a LicensIt Registry, a secured system remote from the viewer. The Registry retains information to validate the credentials of a Tool Box (attempting to register a work) or a LicensIt object, and supplies the remote services and data required by the LicensIt system. The Registry may also supply attribute data obtained indirectly from a content provider's existing legacy database.

By applying object concepts to digital media, NetRights' LicensIt product transforms digital content into an active, helpful participant in the process of rights management. LicensIt objects capable of tapping the various LicensIt support systems become individual, mobile launching pads for new media commerce.

1.2. Requirements by User Domain

The LicensIt system targets the segment of the multimedia industry that produces and sells content elements to CD-ROM and Internet (Web) developers. Therefore, the following comprise LicensIt's customer constituency: creators and owners of digital works, rights administrators and managers, publishers and distributors of digital work, and multimedia developers who compose derivative works.

1.2.1. Content Creators and Owners

From the point of view of the LicensIt system, content creators and owners jointly form a single class of system agent, even though they may be distinct individuals (for example, a commissioned work.) They associate content and other attributes into LicensIt objects. They must:

1.2.1.1. Attach their names and other rights related information (attributes or properties) to a work.

1.2.1.2. Specify aesthetic presentation of the attached information for each work.

1.2.1.3. Be assured that attached information is not easily removed, altered or forged.

1.2.1.4. Be assured that attached information is accessible from any representation of the work, especially from a rendition of the content as well as any iconic ones.

1.2.1.5. Specify an optimal balance between accessibility and locality of LicensIt object properties. Certain static information, such as author, may be located in the envelope, whereas requests for volatile information, such as quantity published, would be referred to a remote server.

1.2.1.6. Specify permissions and requirements for use.

1.2.1.7. Specify other services, such as email, that are available from a LicensIt object.

1.2.1.8. Arrange sets of attributes, presentations, and permissions, and apply them to multiple works. Be able to catalog, share, and generally manipulate such sets in an organized way. (*see 1.2.3.5*)

1.2.1.9. Have available prototypes or templates of sets of attributes, presentations, and permissions that are formally and legally appropriate to various types of work.

1.2.1.10. Have the greatest degree of automated help

1.2.1.10.1. organizing attribution and credit when LicensIt works are included in derivative LicensIt works.

1.2.1.10.2. evaluating, granting, and tracking permitted uses.

1.2.1.10.3. complying with protocols established by various registries. (*see 1.2.3.5*)

1.2.1.11. Have access to packaging process both from within creativity tools and directly from the shell.

1.2.2. Content Users / Multimedia Developers

Multimedia developers compose multiple LicensIt elements and non-LicensIt elements, possibly into a containing LicensIt work.

1.2.2.1. Locate work on the Web

1.2.2.1.1. Search for work by media type, subject, author, owner, and registry.

1.2.2.1.2. Search the domain of all registered LicensIt works available from the Web.

1.2.2.1.3. Obtain search results within response times on the order of 5 seconds.

1.2.2.1.4. Automatically locate alternate versions, formats and renderings.

1.2.2.2. Have no need for LicensIt specific product knowledge to locate or manipulate LicensIt objects.

1.2.2.3. Readily and easily access a rendition of a work for the purpose of evaluation ("try it out").

1.2.2.4. Identify owner, author, and other contributors to a work in the course of evaluation.

1.2.2.5. Be assured of the authenticity, integrity and validity of a work.

1.2.2.6. Enter into communication with rights holders directly from the work, with automatically supplied context (*e.g.*, from, to, re).

1.2.2.7. Enter into transactions for use directly from the work, with automatically supplied context (*e.g.*, from, to, re).

1.2.3. Registries and Rights Administrators

Registries and rights administrators may assume the role of owner for work they control or represent, binding large numbers of works into LicensIt objects. When they do, they will require a separate interface more suitable to batch processing. Also, registries and content servers are frequently not distinct classifications, since registries often store and serve the work they represent.

1.2.3.1. Authorize the use of the registry by particular creators.

1.2.3.2. Administrate (add, delete, suspend, reinstate, bill, and report) accounts under which use is authorized.

1.2.3.3. Associate, organize, and report on registry content.

1.2.3.4. Automate to the greatest possible degree the acquisition of works (additions to the registry).

1.2.3.5. Express in terms of sharable templates (property sets, rule sets) the protocols and information required by the registry. Such templates should be available for transfer and direct use by creator/owners. (*see 1.2.1.8, 1.2.1.10.3*)

1.2.3.6. Fully automate authentication and validation of work (over the Internet).

1.2.3.7. Automate to the greatest possible degree the granting of permissions.

1.2.4. Content Servers and Repositories

1.2.4.1. Provide the ability for any platform to serve content through the registry system. Using LicensIt cannot require re-implementation of a legacy server.

1.2.4.1.1. Must have a stable, published interface protocol.
1.2.4.1.2. Must server both content and meta-data (attributes).

1.2.4.2. LicensIt must include a built in repository system.

1.2.4.3. Support a "distributed" work, where access to content constitutes a transaction. Authorization (subsequent to initial agreement to terms) should be automated.

1.3. Implementation Requirements

1.3.1. Requirements Due to Operating nvironment

1.3.1.1. All implemented protocols must be compatible with the standards, speeds and capabilities currently in use on the Internet.

1.3.1.2. The browsing and creative tools must run on Mac and Windows. The registry must run on Windows NT. The repository must run on Windows NT. The repository API must be supported on Windows NT and Unix.

1.3.1.3. Size and efficiency of the viewer matter greatly because the capabilities of browsing machines will always lag behind the latest technology.

1.3.1.4. All system idioms such as cut, paste, drag and drop, context clicking, etc., should be observed.

1.3.1.5. Where content is transferred out of the LicensIt object, LicensIt attempts to control and record such action. LicensIt should maintain a database of works in progress and their antecedents.

1.3.1.6. LicensIt must support a bulk environment, where large numbers of works are created and/or registered in batch (probably via API).

1.3.2. Requirements Due to Marketing nvironment

1.3.2.1. Low profile. LicensIt objects act just like their raw data counterparts when composed, viewed, moved, cut, pasted, etc. The LicensIt product extends the current environment (such as property pages) rather than present its own interface and modes of operation.

1.3.2.2. Interoperability based on open standards (objects systems) wherever possible, as opposed to "plug-ins" or proprietary APIs.

1.3.2.3. Interoperability with the greatest number of existing and future tools, preferably without explicit cooperation or mutual knowledge between a tool and LicensIt. Interoperability with market leading creativity tools.

1.3.2.4. Ability to contain all common digital formats, such as GIF, TIF, JPG, BMP, WAV, AVI, etc.

1.3.2.5. There must be a distinction between LicensIt and *ad hoc* attributes. Certain aspects of format (*e.g.*, logo is present) and content (*e.g.*, LicensIt antecedents always denoted) should be enforced and/or automated.

1.3.3. Requirements Due to Business nvironment

1.3.3.1. Security of transactions and communications, especially monetary transactions and authentication. Peer review, publication and validation required of all algorithms.

1.3.3.2. Interoperation with major network commerce and security utility providers.

1.3.3.3. All transactions should be automatically reported and routed, and must be auditable.

1.3.3.4. Support legacy systems (such as HTML based ordering) to the greatest possible degree.

1.3.4. Requirements Due to Sales and Distribution nvironment

1.3.4.1. Viewer must be readily, if not automatically available while browsing on the Web.

1.3.4.2. Viewer must be small (efficient) for transfer across slow networks.

1.3.4.3. Viewer must be low/no support, since it will be free, given away in the largest possible quantities, and the user is unaware of its function.

1.3.4.4. Any and all component configurations must be possible, since they cannot be avoided and will naturally occur due to Internet distribution. Full forward/backward, permanent compatibility where features degrade gracefully to those of the least capable component.

2. Systems Analysis

The Systems Analysis is presented in object format. Because of the loose coupling of the components, the system is not described as fully integrated. Rather, it is described as several separate agents, each with a unique system state. These include the Registry, the Desktop, and the Tool Box.

2.1. Classes and Class Relationships (Object Model)

2.1.1. LicensIt Registry

The following are the major components of the LicensIt Registry. (See design diagram xx).

2.1.1.1. Repository

The Repository is a subsystem responsible for supplying LicensIt files (persistence) to client applications such as browsers and custom applications across the Internet. It consists of the object repository handler, the Just-In-Time (JIT) Packager, the Data Service, and the Object and Referred Store databases.

2.1.1.2. Registration and Packaging (LPAPI)

The Registration and Packaging subsystem is responsible for creating LicensIt persistence files (bulk) and registering LicensIt objects created by both the Express Packager (bulk) and the LicensIt Tool Box (remote). It consists of the Template Editor (which creates templates, or patterns of Binders, Page layouts, and Property Sets for LicensIt objects), the Bulk Packager User Interface, the Packager and Object Registration subroutines, and the Bulk, Template, and Registration databases.

2.1.1.3. Object Services

The Object Services subsystem is responsible for providing services to remote LicensIt objects across the Internet. It consists of a Listener and an Object Services manager. The Listener performs Object and User validation, and comprises the System Security database. It activates a processing thread in the Object Services Manager, which calls upon the Registration, Licensing (Contract), Mail, and Property (lookup) services. The Object Services subsystem contains the System Security, contracts, and Billing databases.

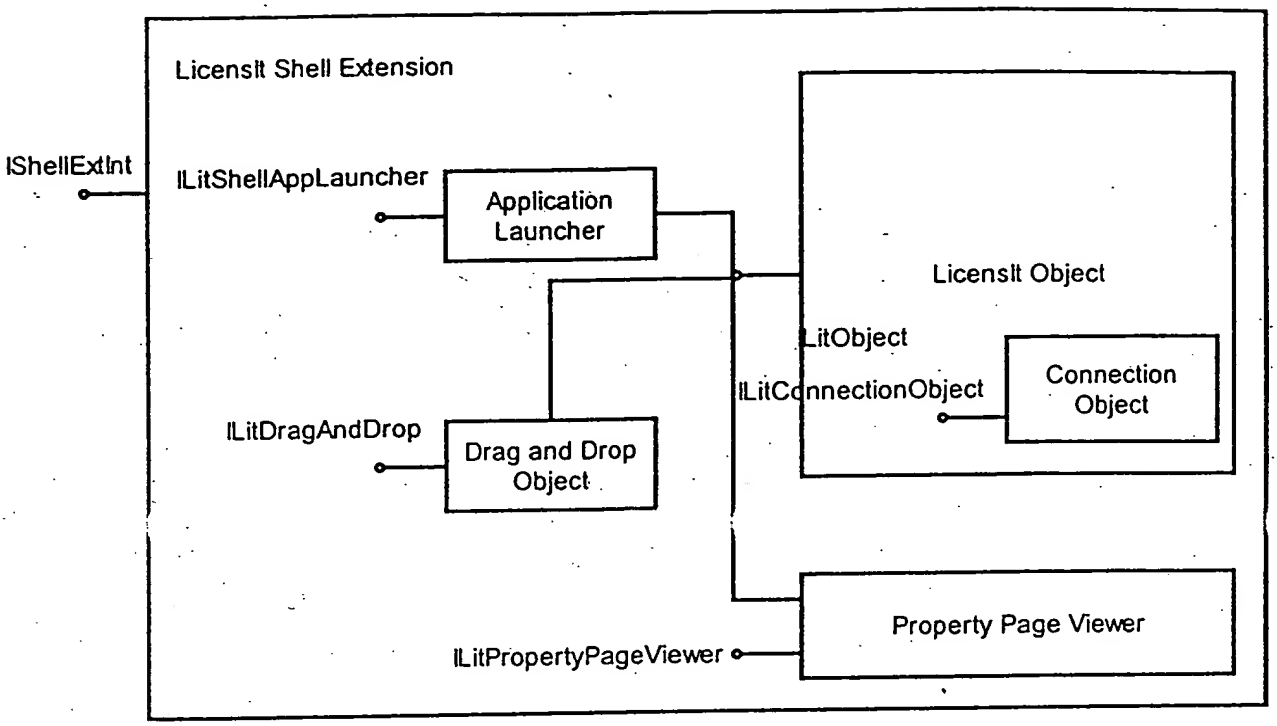
2.1.2. LicensIt Desktop

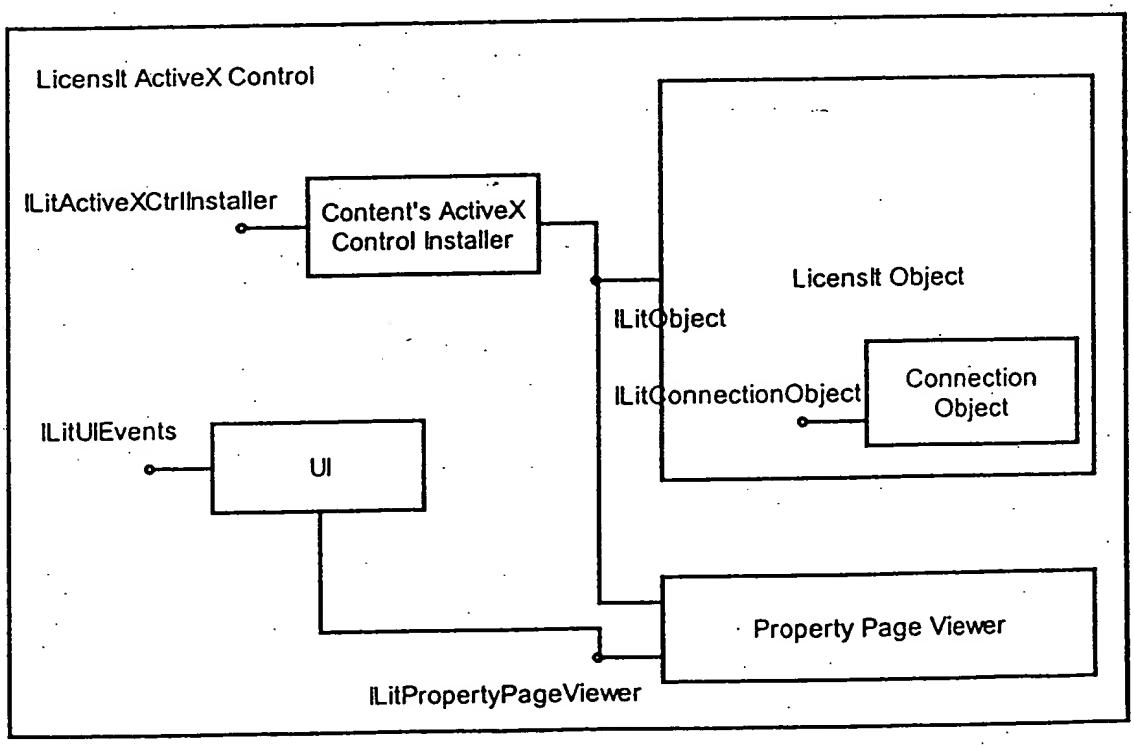
Two logical components — the LicensIt Shell Extension and the LicensIt ActiveX Control — comprise the LicensIt Desktop apparent to the user. The LicensIt Shell Extension is responsible for responding to direct manipulation of the iconic representation of a LicensIt object, such as activation (double click), or property display (context, or right click). The LicensIt ActiveX Control is used to display LicensIt content from within an application, such as within a browser.

These two are themselves composed of several internal component objects. The LicensIt Object provides the base content functionality shared by both the LicensIt Shell Extension and the LicensIt ActiveX Control. The Property Page Viewer provides the platform specific ability to display and animate the properties of the base object. (See section XX for a description of the internal structure of a LicensIt object.)

Initially supported LIT content file types are JPG, GIF, and Bitstream (with no player).

The two figures on the next page show the structure of the two logical components:





2.1.2.1. ActiveX Control

- Embodies the entire functionality of the LicensIt ActiveX Control.
- Besides being an ActiveX Control itself, it is also an Active X Control container. This LIT ActiveX control contains another LIT ActiveX Control that does the actual displaying of the LIT object's content.
- Provides LicensIt altered property pages.

2.1.2.2. Shell Extension

- Embodies the entire functionality of the LicensIt SHELL extension.
- Implements the IShellExtInit OLE Interface. To provide a SHELL extension, an IShellExitInit interface is required by the explorer.

2.1.2.3. Application Launcher

- Extracts the content out of the LicensIt Object.
- Launches an application associated with the content of a LicensIt Object. The application is opened with the contents inside of it.

2.1.2.4. Drag and Drop Object

- Supports drag and drop of LicensIt file icons.

2.1.2.5. LicensIt Object

- Embodies the entire functionality of the LicensIt Object.
- Does NOT provide any UI functionality.

2.1.2.6. Connection Object

- Represents a connection which performs lower level I/O for getting dynamic properties from a particular server(s).
- Persists information, such as, the address(es) of the server(s) associated with the LicensIt object. This information is stored in the LicensIt object's associated .LIT file.

2.1.2.7. Property Page Viewer

- Handles both the display of and interaction with property pages

2.1.2.8. ActiveX Control UI

- Mostly handles the right-mouse for the LicensIt ActiveX contol.

2.1.2.9. Content ActiveX Control Installer

- Checks for the installation of the ActiveX control associated the LicensIt Object's content.
- If necessary, installs, from the internet, the content's ActiveX controller.

2.1.3. LicensIt Tool Box

2.1.3.1. Template

2.1.3.2. Binder

2.1.3.3. Page

2.1.3.4. Property Set

2.1.3.5. Permission

2.1.3.6. Control

2.1.3.6.1. User Interface
2.1.3.6.2. Service

2.2. Operations (Interface Model)

2.2.1. Registry Operations

2.2.1.1. Repository

2.2.1.2. Registration and Packaging (LPAPI)

2.2.1.3. Object Services

2.2.1.4. Redirecting

2.2.1.5. Mailing

2.2.1.6. Storing/Supplying Binders

2.2.1.7. Supplying Properties

2.2.2. Desktop Operations

The following is a list of functions performed by the LicensIt client-side software. *At the end of each entry is a list, in italics, of COM methods that MAY be called in the implementation of a function.*

2.2.2.1. General Object Operations

2.2.2.1.1. Authentication

The LicensIt object calculates a message digest and compares it to the signed message digest in the LicensIt object structured storage (put there at registration). Any recently cached public key from the server is presumed to be correct for this purpose, but may be verified either globally or by being compared to a NetRights signed reply from the server.

ILitObject::verifyDigitalSignature
ILitObject::calculateMessageDigest
ILitObject::decrypt
ILitObject::compareMessageDigest

2.2.2.1.2.Licensing

Obtaining rights to a LicensIt object is done through interaction between a contract control and the LicensIt object. The contract control gets the layout for a contract dialog from the LicensIt object, displays it to the user who supplies any input and presses "generate contract." The contract control displays a textual contract that the user may accept or reject. An accepted contract is sent to the server via the LicensIt object, where it is signed and sent back. The user signature is then added and the object stores the contract and sends a fully executed copy back to the server, where a copy is filed and billing is done.

Any contract that does not have a price attached is deemed to require manual intervention at the server. The user submits — as opposed to accepts — the generated contract, and the process is interrupted with the message that the contract has been submitted. It is incumbent upon the registry to reply with an object that can satisfy the request automatically (*i.e.*, computes a price) at some future date.

ILitObject::enumContractLayouts
ILitObject::readContractLayout
IContract::displayContract
ILitObject::acquireRightsContract
IConnectionObject::submitContract

2.2.2.2. Viewing

2.2.2.2.1.Content

The content may be viewed by either the LicensIt Shell Extension or from an application hosting the LicensIt ActiveX control.

The LicensIt Shell Extension controls how the user interacts with a desktop icon that represents a LicensIt object. The desktop icon looks similar to the content's native application icon, if available, with a LicensIt banner displayed across it. Context-clicking the icon and then choosing properties will activate property display (see2.2.2.2.2). Double clicking the icon will invoke the application of control associated with the content type.

ILitShellAppLauncher::getContentExtension
ILitShellAppLauncher::getApplication
IDataObject::

The shell extension also implements drag and drop.

ILitDragAndDrop::loadOLEClipBoard

The LicensIt ActiveX control itself contains another ActiveX control that displays the LicensIt content. This control bootstraps the remainder of the LicensIt desktop when loaded from a browser (over the Internet).

ILitActiveXCtrlHelper::isCLSIDInstalled
ILitActiveXCtrlHelper::installActiveXCtrl
ILitActiveXCtrlHelper::loadActiveCtrl
ILitActiveXCtrlHelper::runActiveXCtrl

The LicensIt ActiveX control completely covers the display and intercepts all keyboard and mouse input (focus). Right-clicks (context click) invoke the property pages (see 2.2.2.2.2)

ILitUIEvents::

2.2.2.2.2. Properties

Both the LicensIt Shell Extension and the LicensIt ActiveX Control host a LicensIt Property Page Viewer. The Property Page Viewer in turn hosts all controls and displays that interact with the LicensIt object to display meta-data, acquire rights, send email, etc. The general sequence of operation are as follows:

- The Property Page Viewer reads the layout from the LicensIt object.
- The Property Page Viewer reads the properties from the LicensIt object and populates the dialog generated from the layout.
- Individual controls interact as designed with the LicensIt object and, indirectly, the connection object and the server.
- The dismissal of the property pages terminates the Property Page Viewer.

ILitPropertyPageViewer::
ILitActiveXCtrlInstaller::isCLSIDInstalled
ILitActiveXCtrlInstaller::installActiveXCtrl
ILitActiveXCtrlInstaller::loadContentInfoActiveXCtrl
ILitObject::getPropertySet
ILitObject:: (service calls)

2.2.3. Tool Box Operations

2.2.3.1. Defining/Acquiring Templates

2.2.3.2. Packaging

2.2.3.3. Registering

Registration is accomplished through cooperation between the template and the LicensIt object. Additional capabilities afforded to the LicensIt object by the Tool Box are aggregated onto the basic LicensIt object. Principal among these capabilities is the licensing interface.

ILitObject::registerObject

2.3. Data Dictionary

Name	Type	Agent	Arguments	Description
Attribute	Alias			(see Property).
Backstop Address	Agent			The server to be contacted when all other addresses cannot be contacted. This address is contained in each LicensIt Object.
Base Property Set	Class			A required property set. Contains the set of required properties, among them is a Server Search path.

Certificate of Permission	Agent			an electronic document that serves as proof of an executed contract.
Certificate of Use	Alias			(see Certificate of Permission).
Content	Alias			(see Work).
Contract	Class			A certificate of permission, an electronic document, signed by a rights holder and an object user. Either signature may be implicit. The result of a transaction.
Control	Class			A graphical interface element on a Property Page that displays a property value. The control also contains a Map to a Property Descriptor used to find the property(ies) that are displayed in the control. (May be used for different properties, on different property pages or elsewhere on the same property page.)
Desktop	Agent			The system used to view content/works. Consists of a PC and operating shell, and may contain browser.
Forwarding Address	Agent			address of where to send a transaction if an address cannot be resolved.
Home	Agent			the Registry that registered the current LicensIt Object.
Licenser	Agent			Registry proxy for owner of rights.
LicensIt Object	Class			A Template that contains graphical layout information and controls to display properties, and a Property Set that defines the properties and content (work).
LPAPI	Agent			The LicensIt Packaging API. The Application Programming Interface used by the LicensIt Creator's Tool Box and the LicensIt Express Packager to create and optionally register LicensIt Objects.
Map	Class			The object stored in a control that contains the information that points to a Property Descriptor ID (to reference a Property) to be displayed in the control, or to a Method to be executed when the control is activated.
Minimum Permission	Agent			actions allowed on an object without external contact; (i.e. without acquiring a contract).
Permission	Agent			the process of communicating with a Licenser (at a Registration Server) and securing a contract.
Predicate	Agent			a prerequisite of a statement. Proposed business deal.

Pricing expression	Agent			algebraic statement executed to compute a price for a particular deal.
Property	Class			An assigned, named metadatum or datum within a LicensIt Object that contains a rendition of the work or some characteristic of the work; e.g., Content (the 16 bpp version), Author, Title, etc.
Property Desc	Class			A label, data type, size, and edit rule describing a property.
Property Page	Class			A collection of controls displayed together (in a tabbed dialog).
Property Set	Class			A list or tuple of (group of) properties or named values stored together or as a single record and associated with a work. E.g.. "Author, title, publisher".
Registry	Agent			The system used to register, locate, authenticate, and interface with LicensIt objects. A registration server.
Repository	Class			Server Service to provide LicensIt files to applications over the network (e.g.. browsers, custom applications, etc.)
Template	Class			A collection of Property Pages and a Property Set that defines the composition of a LicensIt Object.
Tool Box	Agent			The system used to create LicensIt objects and register them.
Work	Class			The (required) content property of a LicensIt Object; the "payload".

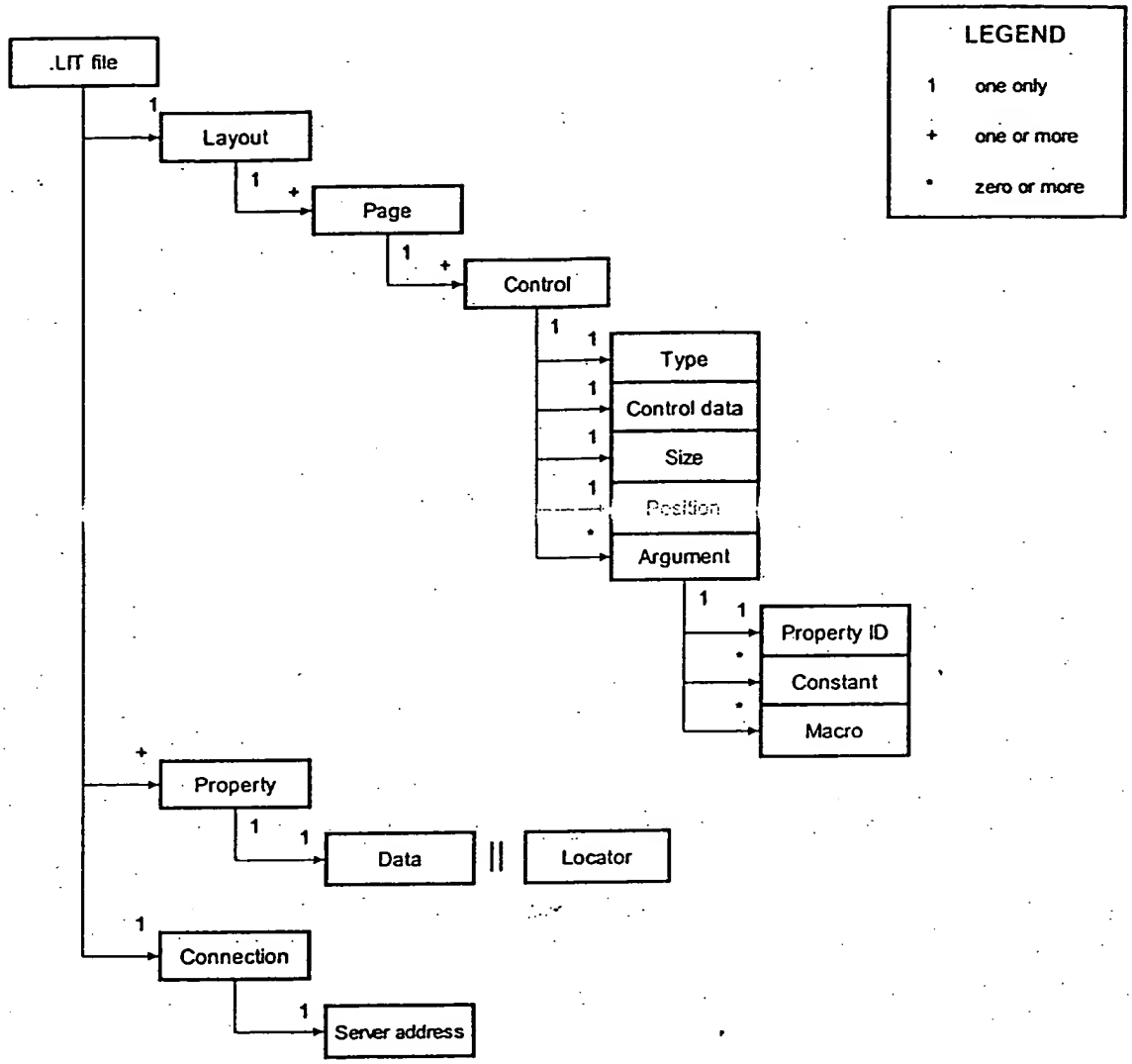
3. System Design

3.1. Data Design

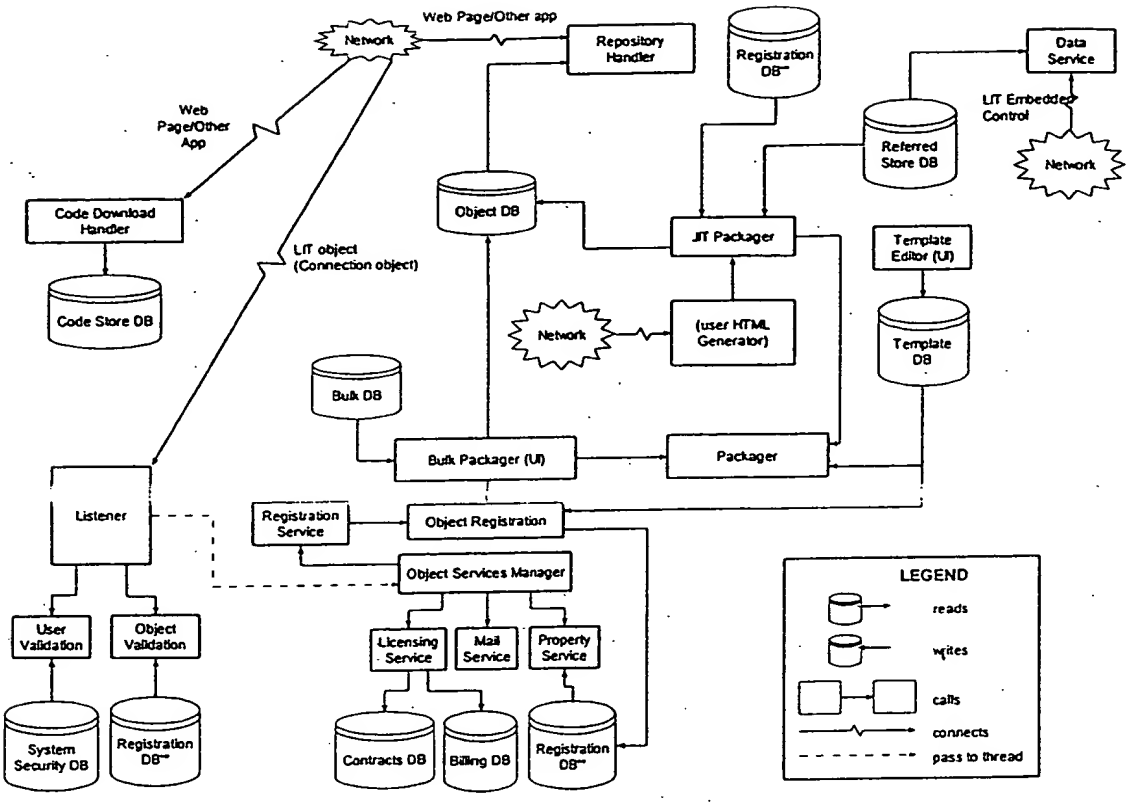
3.1.1. Data Design Dictionary

Name	Type	Arguments	Description
Billing	Database		Calculated costs to clients
Bulk	Database		Input to Bulk packaging
Bulk Packager UI	Interface		Operator interface to run bulk packaging
Code Download Handler	Routine		Respond to request for LicensIt code. (HTTP?)
Code Store	Database		LicensIt code satisfaction
Contracts	Database		Stored contracts executed from LicensIt objects
Data Service	Routine		Respond to requests for unpackaged data (e.g., for a URL property in an HTML control)
JIT Packager	Routine		On-the-fly object request satisfaction (creation)
Licensing Service	Routine		Sign and store contracts, generate billing
Listener	Routine		Accept connections and route them by new or existing (registered) object. "Pass" connection to a thread in either Register or Object Services Manager process.
Object	Database		Stored, complete LIT files (if any)
Object Registration	Routine		Store object metadata
Object Services Manager	Routine		Switch for object services (calls appropriate one)
Object Validation	Routine		For existing object, look up and either validate, invalidate (for objects that are 'positively' unregistered), or bounce to correct server.
Packager	Routine		API to generate a LicensIt storage (OLE Structured)
Property Service	Routine		Satisfy requests for (meta-) data (properties)
Referred Store	Database		Directly accessible (indirect) data; usually content (e.g. AVI files)
Registration	Database		Metadata base
Repository Handler	Routine		Locator and supplier of LicensIt objects
System Security	Database		Validation of users of registry
Template	Database		Stored Templates
Template Editor UI	Routine		Visual interface for creating templates
User Validation	Routine		For new object, look up registering party (validate)

3.1.2. Object


LEGEND
1 one only
+ one or more
* zero or more
.LIT file
Layout
Page
Control
Type
Control data
Size
Position
Argument
Property ID
Constant
Macro
Property
Data
Locator
Connection
Server address

3.1.3. Registry



3.1.3.1. Registration Server Component Design

Listener

The purpose of the Listener is to respond to connection requests from the Connection object contained within a LicensIt object. The Listener accepts connections using a known port and protocol. After validating a LicensIt object and its user, the Listener passes the connection to a thread on the Object Services Manager. Once the connection has been passed, the Listener is no longer involved and the LicensIt object interacts directly with the Object Services Manager.

In order to validate a LicensIt object and its user, the server address of the LicensIt object's Registration Server is passed from the Connection object to the Listener. The server address is used to map an ODBC data source name for the Registration database. The Listener validates a LicensIt object by opening an ODBC connection to the Registration database and making a query using the object's unique identifier. If the database returns a record for the object, the object is considered valid. A user is validated by the Listener opening an ODBC connection to the System Security database and making a query using the user's unique identifier. If the database returns a record for the user, the user is considered valid.

If the LicensIt object and its user are valid, the Listener passes the connection to a thread on the Object Services Manager. If either the LicensIt object or its user are determined to be invalid, the connection is closed and no thread is started on the Object Services Manager.

Object Services Manager

The purpose of the Object Services Manager is open threads for connections passed from the Listener. The connections passed are from a Connection object contained within a LicensIt object. When a LicensIt object requests a particular service, the Object Services Manager calls the appropriate service via a Connection object. The available services are:

Registration Service (Registering an object)

License Service (Licensing an object)

Property Service (get/set an object's properties)

Mail Service (send an email if such service not available on the client)

Registration Service

The purpose of the Registration Service is to register an unregistered a LicensIt object. When a LicensIt object attempts to register itself, it sends the request to the Object Services Manager, which calls the Registration Service. The Registration Service calls Object Registration which makes an insert query by opening an ODBC connection to the Registration database.

Licensing Service

The purpose of the Licensing Service is to license a LicensIt object to a user based on agreed terms and price. When a user attempts to license a LicensIt object, the LicensIt object creates a contract and sends it to the Licensing Service. If the contract contains a price, the Licensing Service automatically signs the contract and returns the contract text to the user for acceptance/rejection. If the user accepts the contract, the LicensIt object makes a second request to the Licensing Service. On the second request, the Licensing Service verifies the user's credit card, writes the contract to the Contracts database, writes an invoice to the billing database and returns a copy of the contract to the user (signed by both parties).

Property Service

The purpose of the Property Service is to provide a LicensIt object access to its dynamic properties. A LicensIt object gets or sets its properties making the appropriate request to the Property Service. The Property Service opens an ODBC connection to the Registration database makes a select query for property inquiries and an insert or update query when modifying properties.

Mail Service

The purpose of the Mail Service is to provide a mail transport from a LicensIt object to the creator/owner of the object. In cases where the user may not have an email service available, a LicensIt object will send a request to Mail Service. The LicensIt object will pass the email message along with the request and the

Mail Service forwards the email message to the creator/owner by using the Simple Mail Transport Protocol (SMTP).

Object Registration

The purpose of Object Registration is to register a LicensIt object with a particular Registration Server. The Object Registration receives a request from the Registration Service to register a LicensIt object. A template is read from the Template database in order to determine the properties to be stored for the object. The Object Registration then opens an ODBC connection to the Registration database and inserts records which represent the properties of the LicensIt object.

Repository Handler

The purpose of the Repository Handler is to return a LicensIt object's persistence (.LIT file) to a calling web page or application. The Repository Handler is contacted using a known port and protocol.

Packager

The purpose of the Packager is to create the persistence of a LicensIt object (.LIT file). The Packager is called from either the Express Packager or the JIT Packager. The Packager reads a template from the Template database to determine properties of the object. The property information is combines with information supplied by the Express Packager or JIT Packager to create a .LIT file. The .LIT file(s) are then returned to the process which requested packaging.

Bulk Packager

The purpose of the Express Packager is the creation of large numbers of .LIT files by using bulk data. . The bulk data is stored on a Bulk database and passed to the Packager which creates the .LIT file. The created .LIT files are then returned to the Express Packager and stored in the Object database. The Express Packager operates from its own user interface.

Just-In-Time (JIT) Packager

The purpose of the Just-In-Time (JIT) Packager is to create a .LIT file upon request for user-generated HTML. The JIT Packager first checks the Object database for a previously created .LIT file. If it exists and it is a JIT created file, the JIT date/time stamp is updated. At this point, if a .LIT file exists, no further action is taken. Otherwise, The JIT Packager opens an ODBC connection to the Registration database to retrieve the object's properties, reads other referred data from the Referred Store database, and combines them to create a .LIT file, which it stores in the Object database. Such .LIT files are marked as JIT files with a date/time stamp. .LIT files with JIT date/time stamps more than a certain age are periodically purged from the Object database.

Data Service

The purpose of the Data Service is to respond to requests for unpackaged data for an embedded LIT control. The Data Service receives requests from a LicensIt Embedded Control using a known port and protocol. The unpackaged data is read from the Referred Store database and loaded into the Embedded LIT control.

Code Download Handler

The purpose of the Code Download Handler is to respond to requests for LicensIt code from a web page or other application. The requests for LicensIt code are received using a known port and protocol. If the request can be fulfilled, the LicensIt code is read from the Code Store database and returned to the web page/app.

3.2. User Interface Design

3.2.1. Registry

3.2.1.1. LicensIt Template/Template Editor

The LicensIt Template contains the blueprint for an individual LicensIt Object. It defines each of the properties as well as each of the Property Pages used to create and display a LicensIt Object. It is the Template that defines the structure of a LicensIt Object.

The Template is created using the **LicensIt Template Editor**. The LicensIt Template Editor produces the file that is used as input to the LicensIt Creator's Tool Box and the LicensIt Express Packager and by custom applications utilizing the LicensIt Packaging API (LPAPI). Minimally, the Template contains the following entities:

Content Descriptor
- Aspect
  - Allowed
  - Required
- Rendition
  - Allowed
  - Required

Metadata Descriptor
- Property Descriptor ID
- Location
- Required

Property Pages(s)
- Controls
- Dialog Stuff (x, y position, width, height, control type, label, etc.)
- Map
  - Property Descriptor ID
  - Edit Rule

Property Set
- The Property Set contains the definition of each property that is contained in the LicensIt Object.

When a template is used as input to the LicensIt Express Packager, an input data file that defines individual properties for each of the LicensIt Objects is required. A skeleton input data file that specifies

the format that must be followed when creating the input data file can automatically be generated by LicensIt Template Editor based on the controls and properties defined in the template.

3.2.1.2. LicensIt Express Packager

The LicensIt Express Packager will provide the means of creating one or more LicensIt Objects in a batch or unattended mode. Input to the Express Packager will be a pre-existing template and an input data file.

The LicensIt Express Packager will call the LPAPI to read and translate the template into objects and/or data structures that are used by the Express Packager as a blueprint for a LicensIt Object. The LicensIt Express Packager opens and reads the input data file and pairs input data with a property from the template. When the contents of an entire LicensIt Object is read from the input data file and paired with a property, the LPAPI is called to create the LicensIt Object. Optionally, an additional LPAPI call will be made to register the object with a registration server.

The input data file will specify where the content is coming from, and where it should be (or is) stored. The input data file will specify one of the following:

- the Content should be included in the LicensIt Object
- the Content should be moved onto the LicensIt Registration Server
- the Content exists in a legacy system
- the Content exists in a file system

3.2.1.3. LicensIt Server

3.2.1.4. LicensIt Download Services

3.2.2. Desktop

3.2.2.1. LicensIt Object

3.2.2.2. LicensIt Control

3.2.2.3. LicensIt Extension

3.2.3. Tool Box

A LicensIt Object may be created using either the **LicensIt Creator's Tool Box**, the **LicensIt Express Packager**, or via a custom, user created application. All methods of creating LicensIt Objects use the common **LicensIt Packaging API**, henceforth, **LPAPI**, to actually create the object and optionally register it with a **LicensIt Registration Server**.

LicensIt Objects may be created and optionally Registered with a LicensIt Registration Server, as stated above. This will yield a LicensIt Registered Object. If a LicensIt Object is not registered with a registration server, the object is said to be **Unregistered**. Both types of objects are supported:

A Unregistered LicensIt Object must contain all of the properties inside the object itself; in local storage.

A Registered LicensIt Object allows storage of properties in either the LicensIt Object, on a LicensIt Registration Server, or at both locations.

A LicensIt Object, once created, will contain a global unique identifier that is assigned by the LicensIt API as part of the creation process.

The LicensIt Creator's Tool Box provides a Graphical User Interface (GUI) application used by an author to create LicensIt Objects. Each Creator's Tool Box will be provided with a unique digital signature (which may take the form of a software registration number). The Creator's Tool Box must be registered with a LicensIt Registration before it can be used to create registered LicensIt Objects.

The Creator's Tool Box will allow creation of a LicensIt Object based on a pre-existing template. The UI of the Tool Box will allow the author to select a template and call the LPAPI to read and translate the template into objects and/or data structures that are used by the Tool Box to interpret the object. The Tool Box then creates a tabbed dialog containing each of the property pages so the author can traverse the dialog and fill in the data. It should also be noted that the author will have the option of filling information once and saving it so that future templates can be pre-filled.

When the author has completed filling in the properties, the object is ready for creation and registration. Each of the fields of the tabbed dialog will be read, filling in the appropriate data structures. The data structures will be passed to the LPAPI to be used to create the LicensIt Object. A separate, additional API call will be made if the LicensIt Object should be registered with a LicensIt Registration Server. Calling the LPAPI to create the object will result in a File Open dialog to be called to allow the user to specify an output filename for the new LicensIt Object.

3.3. SDK/API

4. Implementation and Methodology

This section describes some of the technologies available for implementing the LicensIt product. Included are discussions of the pros and cons of each technology. Since selection of the correct implementation tools and techniques is vital to the success of the product, the focus is on the key decisions which need to be made.

4.1. Object Design

Since before the beginning of time (relative to NetRights, or Fall, 1995), it was decided that the digital works that LicensIt would manage should be treated as generic "objects". This enables the leveraging of operating system features and extensions, rather than the re-invention additional proprietary file formats. It also means that instead of requiring custom modifications to the myriad commercial applications which currently manage digital works, the LicensIt attribution information can be carried in a native format already understood by them.

Unfortunately, the object technologies are not as advanced as we would like. There are currently two competing standards evolving, and both of them are relatively early in their adoption cycles. The two standards that could reasonably be used in the LicensIt products are Microsoft's OLE (Object Linking and Embedding), recently renamed "ActiveX", and OpenDoc, developed by CI Labs, a consortium led by Apple and IBM.

4.1.1. OL /ActiveX

ActiveX (formerly named OLE) is Microsoft's technology for component architecture. It was originally developed as a mechanism to allow sharing of documents in the Microsoft Office application suite (Word, Excel, PowerPoint and Access). ActiveX uses the Component Object Model (COM) architecture as the primary interface mechanism. COM has recently been expanded to support distributed objects, extending the object communications boundary from processes within a machine to processes anywhere on a network.

With the rapid growth of the Internet, Microsoft has steered its strategy to include TCP/IP and the related Internet protocols (HTTP, FTP, SMTP, etc.) within the ActiveX framework. The latest pre-release version of Internet Explorer for Windows 95 includes support for ActiveX objects embedded in web pages.

The current LicensIt prototype is implemented under Windows 95 using ActiveX technologies, and for the Windows platforms, this would appear to be a logical choice for completion of the final product. However, there are a number of concerns with this strategy:

- The ActiveX technology is owned solely by Microsoft. No committees or consortiums are available to steer the standard in ways that may not benefit Microsoft directly.

- ActiveX is not widely in use (and not fully implemented) on other platforms, especially the Apple Macintosh, which is a primary platform used by a large segment of LicensIt's target market. Macromedia is supposed to be working with Microsoft to keep ActiveX on the Mac in step with the Windows version, but there are no announced delivery dates as of May 1996, and there is a general reluctance in the Mac user community to adopt Microsoft solutions. This reluctance extends to key Mac software development companies such as Adobe, who have not embraced ActiveX in their own applications.

- Distributed COM (DCOM) has only been released in a first beta release for Windows NT 4.0, and will not be generally available until late 1996 or early 1997. The impact of the

availability of DCOM affects the communications with the registration and/or repository servers.

- The current LicensIt prototype uses Microsoft's Foundation Class (MFC) libraries, and as such requires a substantial code component on the client machine (over 1MB). This is fine for an application installed from diskettes or CD-ROM, but excessively large when being downloaded over the Internet in order to be able to view an object in real time. Microsoft is addressing this by creating a "lightweight" ActiveX framework, but it is unclear whether LicensIt will be able to take use it.

4.1.2. OpenDoc

In 1989 the Object Management Group (OMG), a consortium of object vendors, began working on a specification for an object bus, named CORBA (Common Object Request Broker Architecture). CORBA 2.0 was released in late 1994, and OMG has subsequently released a series of object services to layer on top of CORBA. These services include transactions, externalization, query, licensing, etc. Additional services, such as security, are due to be released in 1996.

Ci Labs was founded as a consortium, primarily led by Apple and IBM, to develop a software component architecture around the CORBA standard. The specification is called OpenDoc, and is currently in its first release, available in beta on the Mac and in April 1994 was release in final form for IBM's OS/2 Warp. This implementation of OpenDoc depends on IBM's System Object Model (SOM) for the ORB (Object Request Broker) layer. IBM has extended SOM to support distributed objects with its Distributed SOM (DSOM) architecture.

OpenDoc is roughly parallel to ActiveX in providing software component and object support to traditional personal computer applications. Apple is supposed to be including OpenDoc as a core technology in its next major release of the Mac operating system, code-named Copland. They have also released a beta version of an component-based Internet operating environment, named CyberDog. At this time (May, 1996), CyberDog is one of the few frameworks for developing and testing OpenDoc parts (components).

As with ActiveX, there are a number of major issues concerning using OpenDoc as LicensIt's core object technology:

- The implementations are just barely being made available in their initial 1.0 versions. ActiveX, on the other hand, is nearing its third major release.

- The Windows version of OpenDoc, being developed by IBM, has not even been made available in a pre-release form. IBM promises to have it in developer's hands by the end of May 1996.

- Microsoft and other Windows applications developers are making no commitments to support OpenDoc.

- Major Mac software developers, especially Adobe, are making no firm commitments to support OpenDoc.

IBM and Apple have both acknowledged that they are working on Component Glue, a technology to allow transparent support of ActiveX from OpenDoc parts, and vice-versa. This technology appears to be based on work done by WordPerfect for its applications suite. The technology should be released late in 1996, but there is no guarantee that it will fill LicensIt's needs.

4.1.3. Hybrid Solutions

Since both ActiveX and OpenDoc carry associated risks, it may be necessary to implement LicensIt using one of a number of possible hybrid solutions. Some of the options are:

- Use ActiveX structured storage for the actual LicensIt objects. Use ActiveX for managing the objects on Windows platforms, and use OpenDoc parts with ActiveX code for reading the structure storage on the Mac (and possibly other platforms, such as UNIX or OS/2).

- Use OpenDoc structured storage (BENTO) for the LicensIt objects. Use OpenDoc on all platforms to manage the objects. Component Glue technology on Windows should allow applications acting as ActiveX containers to use the LicensIt OpenDoc parts.

- In addition to the above two options, the Mac may require custom support for those applications that support neither ActiveX nor OpenDoc, such as Adobe's Photoshop. In this case, LicensIt "plug-ins" may need to be developed specifically for selected applications. These plug-ins will still need to be able to read the native structured storage object format, and therefore will most likely include some ActiveX or OpenDoc code.

4.2. Communications

The LicensIt system requires communications between the LicensIt objects and the LicensIt registration (and possibly repository) servers. This communications is vital for a number of reasons, including providing timely and updated attribution information, and managing and performing the licensing process. The licensing process, in addition, may require additional communications to accomplish electronic payment transactions.

There are two primary schemes for implementing the LicensIt communications requirements. They are a native custom TCP/IP protocol or the transparent communications provided by the distributed object technologies (ActiveX/COM or OpenDoc/DCOM).

Because of the dominance of the Internet as a global networking infrastructure, the TCP/IP protocol has emerged as the most important communications protocol today. Other protocols such as IPX, SNA, AppleTalk, async (XMODEM, Kermit, etc.) and other lesser known protocols, could be utilized and supported in the LicensIt architecture, but multi-protocol implementations add a significant amount of complexity to the development, installation and support efforts. Choosing TCP/IP as the initial, and most likely sole, underlying communications protocol should not impede the sales and marketing of the product in any significant way.

The TCP/IP protocol is a relatively low-level protocol. It provides error-free point-to-point data transmission, as well as some basic network addressing. It does not, however, provide other important features such as security or an object-oriented interface methodology. For this reason, the TCP/IP protocol will need to be extended with the application support needed by the LicensIt components. This can be accomplished with custom encapsulation of TCP/IP, or using the services provided by the component object technologies mentioned above.

4.2.1. A Custom TCP/IP Applications Protocol

The safest approach to the communications protocol implementation would be to create a LicensIt-specific interface layer on top of the traditional TCP/IP protocols. This would entail encapsulating session establishment, protocol states, data formats, etc., in such a manner that the LicensIt components would only need to use a set of simple application-oriented classes to communicate with each other.

The reason this approach is the safest is that all it requires is access to the underlying TCP/IP APIs, provided (and stable) on all the target platforms being considered. The Windows platforms, in particular, use the WINSOCK standard, available on all versions of Windows (3.x, 95, NT), and in widespread use today. The downside to a custom TCP/IP solution is that a significant amount of support code needs to be created from scratch.

4.2.2. ActiveX/DCOM and OpenDoc/CORBA

The alternative to implementing a custom TCP/IP protocol is to use the distributed object framework itself. For ActiveX, this is accomplished through the use of DCOM and currently for OpenDoc, through IBM'S DSOM. The LicensIt prototype was developed using a precursor to DCOM that was delivered by Microsoft with the Visual Basic 4.0 Enterprise Edition, and named Remote OLE Automation.

This scheme is elegant, since it is such a natural extension to the tools being used for the LicensIt application code itself. The actual session management and translation and transmission of data is done transparently and without communications code in the LicensIt system. This could reduce development time considerably.

The big problem with this approach is that the DCOM and DSOM technologies are not fully implemented on the desired platforms. DCOM has just been released in a beta for Windows NT 4.0, and won't be available for Windows 95 until late in 1996, at the earliest. It may never be available for Windows 3.x. Microsoft itself has not made any delivery commitments to DCOM on the Mac or major UNIX platforms, although Macromedia and Software AG are supposed to be working on parallel implementations on the alternate platforms. DSOM is initially available for OS/2, but there are no delivery dates or firm plans for the Mac or other platforms. If either of these technologies is chosen to provide the key object/server communications services, then a careful cost/risk analysis will have to be made.

4.3. Database

The LicensIt registration and repository servers are both inherently database applications. While it would be tempting to look at current Object Oriented Database Management Systems (ODBMS) to complement the architecture of the rest of the system, there are a number of factors that necessitate a look at older relational database engines. The two most important of these are the cost and availability of the database engines, and the need to interface to existing databases provided by large customers.

The LicensIt registration server is most likely going to be implemented using the three-tiered business application model, rather than using older strict client-server methods. The three-tiered model breaks the application down into three distinct layers. The client in the three-tier model behaves very much like in a classic client-server model. The client usually provides the user interface to the user, and represents the ultimate destination of the data. The middle tier represents the business process server, and the third tier is the database server. The server has been broken into two major components so that each can be implemented using optimal technologies. For example, the business process server may be written in C++ using object-oriented development tools, while the database engine might be a commercial DBMS running on a separate platform.

The need to interface to existing customer databases provides strong incentive to use the three-tiered architecture. While there may be a LicensIt registration server sold as a stand-alone system running one platform, there will also be a need for a solution with the LicensIt server on one platform and the database on another (for example, and Oracle system on a UNIX machine). It is imperative that the middle tier (the business process) of the LicensIt server (or servers) have flexibility in where and how it accesses the supporting data.

4.3.1. Traditional Database Management Systems

A large number of the commercial business applications available today are based on one of a relatively small number of established database engines. Some of the engines run on multiple platforms, but each platform typically has preferred database systems. Examples of platform/DBMS combinations in widespread use are:

Platform	Database Management Systems
UNIX	Oracle, Informix, Ingres
Windows NT	SQL Server, Sybase, Access 95
IBM Mainframe	DB2

The biggest problem with these database systems is that they are not standardized, either in their APIs or in their SQL implementations. However, it is going to be necessary to interface to them at some level, so a method for hiding the engine-specific details from the LicensIt application code will be needed. One technique for accomplishing this is careful modularization of the application. Another is described in the section "Hybrid Solutions".

Some of the benefits of the traditional DBMS solutions are the cost and robustness of the systems. Access 95, for example, can be distributed with an application with no per-copy runtime royalties. Many of the systems, like Oracle's, have been available for years, and have many revisions and a long track record supporting mission-critical applications.

4.3.2. Object-Oriented Database Management Systems

For the last few years, the Object Database Management Group (ODMG) has been working on specifications for an object-oriented database management standard. Version 1.2 of the ODMG standard was released in December, 1995, and a number of vendors have implemented systems that conform to this standard. However, most of these vendors are small, and do not have track records or installed customer bases that are comparable to vendors like Oracle or Informix. Their solutions are typically more complex and more expensive that the traditional relational DBMSs. It is also difficult to find engineers with skills working with a particular system, whereas many developers have experience working with one of the major SQL-based engines.

The one ODBMS vendor that has established itself and is more well-known is Poet Software. Their technology, Poet, has been available since 1991 and is in its third major release. Version 4.0 has been announced and is supposed to be available in May, 1996. It runs on all major platforms, including Windows 3.x/95/NT, Mac, and UNIX. It supports OLE, ODBC and multi-threading, and conforms to the ODMG 1.2 C++ Binding specification.

The issues with POET, as with the other ODBMS, include cost and interoperability with the other traditional DBMS engines. If a pure ODBMS is considered for use in the LicensIt application, POET should probably be reviewed first.

4.3.3. Hybrid Solutions

There are a couple of options that leverage the object-oriented implementation benefits while retaining the robust DBMS engines. They consist of C++ class "wrappers" and libraries used to encapsulate the DBMS-specific APIs. There is a company, Rogue Wave Software, that produces a commercial class library that interfaces to all the popular DBMS engines. The class library works with most C++ compilers, including any of those on the current target platform list (Mac, Windows, UNIX). This class library might offer the best trade-off between reduced development time and costs (no per-copy runtime licensing), and usage of and interface to the common database engines.

Another alternative that would require more development time, but would provide an optimized object interface, would be to create a custom LicensIt database class library. This library could be developed to work via ODBC from the business process server platform (initially NT, and maybe moving to UNIX), enabling it to plug in to almost any existing DBMS. ODBC interfaces are currently available from all the major vendors; even Poet is providing an interface to their ODBMS.

4.4. Security

Security technology needs to be used for three major purposes in the LicensIt system: They are the authentication of the users and servers, the authentication of the integrity of the digital works or data being encapsulated by LicensIt, and the security and authentication of the licensing transactions.

Some initial research has been done into available security technologies, with the goal of deciding what level of security is appropriate for an application such as LicensIt. There are many basic algorithms that accomplish the various security tasks (encryption, integrity checking, etc.), and there are pros and cons to most of them. In any case, the algorithms are very complex, and it does not make sense to "reinvent the wheel" in this area. Existing code should be licensed and incorporated into the LicensIt system.

In March, 1996 Robert Morris, Sr., an ex-NSA cryptographic expert and computer science pioneer, was hired to review a basic approach to application security for LicensIt. He blessed the scheme described below, although a lot of details still need to be ironed out.

In short, all of the security routines to be used in LicensIt will use the current "public key" algorithms, such as those developed by RSA, a leading commercial supplier of security software. Similar technology is currently in use in the popular Pretty Good Privacy (PGP) software program. Some of these algorithms are patented, and appropriate licensing is going to be necessary. The costs of this licensing, as well as possible alternatives, have not been explored as of May, 1996.

4.4.1. User and Server Authentication

In the LicensIt business model, it is important that both the user and server involved in any transaction be identified. If a content creator is packaging a digital image and registering the resulting LicensIt object with a registration server, the server must know that this is in fact the authorized user. If a publisher wishes to license a work, it is critical that both parties in the electronic contract are legally identified.

The technology for authentication that will be used in LicensIt is the "digital signature". Digital signatures consist of encrypted certificates that can be used to convincingly identify a party. Digital signatures on documents ensure the following (taken from "Advanced Cryptography, 2nd Edition, by Bruce Schneier - the authority):

1. The signature is authentic. The signature convinces the document's recipient that the signer deliberately signed the document.

2. The signature cannot be forged. The signature is proof that the signer, and no one else, deliberately signed the document.

3. The signature is not reusable. The signature is part of the document; an unscrupulous person cannot move the signature to a different document.

4. The signed document is unalterable. After the document is signed, it cannot be altered.

5. The signature cannot be repudiated. The signature and the document are physical things. The signer cannot later claim that he or she didn't sign it.

5. Appendix A: Client-Side COM Objects and Interfaces

NOTE:

The desktop LIT file right-click popup menu includes the following menu items:

Property	menu item and sub-menu item of LicensIt
Authenticate	sub-menu item of LicensIt
MailContentProvider	sub-menu item of LicensIt

To be done with this document:

Select names for Interfaces that describe their purpose as opposed to just adding an "I" to the object's name, when reasonable to do so.

5.1.1. LIT ActiveX Control

- Embodies the entire functionality of the LicensIt ActiveX Control.
- Besides being an ActiveX Control itself, it is also an Active X Control container. This LIT ActiveX control contains another LIT ActiveX Control that does the actual displaying of the LIT object's content.
- Provides LicensIt altered property pages.

5.1.2. LIT SH LL extension

- Embodies the entire functionality of the LicensIt SHELL extension.
- Implements the IShellExtInit OLE Interface. To provide a SHELL extension, an IShellExitInit interface is required by the explorer.

Interface	Method	Comment
IshellExtInit		
		property menu-item selection, by user, invokes ILitProperyPageViewer::
		user dragging and dropping invokes ILitDragAndDrop::loadOLEClipBoard
		authenticate menu-item selection, by user, invokes ILitObject::verifyAuthenticity
		user double-clicking desktop icon invokes ILitSHELLappLauncher::getApplication

5.1.3. Application Launcher

- Extracts the content out of the LicensIt Object.

- Launches an application associated with the content of a LicensIt Object. The application is opened with the contents inside of it.

Interface	Method	Comment
IlitSHELLappLauncher		
	getContentExtensionAndMIME(...)	
	getApplication(content_extension)	
	extractContent(aspect, rendition, platform)	Returns IDataObject for particular aspect, rendition, and platform

5.1.4. LIT drag and drop

- Supports drag and drop of LicensIt file icons.

Interface	Method	Comment
IlitDragAndDrop		
	loadOLEClipBoard(...)	OLE Uniform Data Transfer

5.1.5. LIT object

- Embodies the entire functionality of the LicensIt Object.
- Does NOT provide any UI functionality.

Interface	Method	Comment
IlitObject		
	init(LIT_FILE)	read data from .LIT file
	getContractingInfo(...)	get any signed contracts
	acquireRightsContract(...)	
	getProperyPageLayout(...)	
	getPropertySet(...)	
	putPropertySet(...)	
	register(...)	
	verifyAuthenticity(...)	
	getContentCLSID(aspect, rendition, platform)	if any, returns CLSID for particular aspect, rendition, and platform
	calculateMessageDigest(...)	

5.1.6. Connection Object

- Represents a connection which performs lower level I/O for getting dynamic properties from a particular server(s).
- Persists information, such as, the address(es) of the server(s) associated with the LicensIt object. This information is stored in the LicensIt object's associated .LIT file.

Interface	Method	Comment
IlitConnectionObject		Represents a connection which performs lower level I/O for getting dynamic properties Persists to .LIT. Stores name of server(s) on behalf of a particular project Low level signing of messages between server and LicensIt object

5.1.7. Property Page Viewer

- Handles both the display of and interaction with property pages

Interface	Method	Comment
ILitPropertyPageViewer		Reads and create property page UI

5.1.8. UI

- Mostly handles the right-mouse for the LicensIt ActiveX contol.

Interface	Method	Comment
IlitUIevents		traps right-click and adds LIT property pages

5.1.9. Content's ActiveX Ctrl Installer

- Checks for the installation of the ActiveX control associated the LicensIt Object's content.
- If necessary, installs, from the internet, the content's ActiveX controller.

Interface	Method	Comment
IlitActiveXCtrlInstaller		
	isCLSIDinstalled(CLSID)	
	installActiveXCtrl(CODEBASE,CLSID)	
	loadContentIntoActiveXCtrl(...)	
	runActiveXCtrl(...)	

6. Appendix B: Time stimates

6.1.1. Client-Side COM objects

Component	Person Days (aggressively estimated)
LIT object	15
Connection object	6
Property Page Viewer	10
LIT drag and drop	5
Application Launcher	8
UI	2
Content's ActiveX Control Installer	5
LIT Shell Extension (excluding external COM objects)	8
LIT ActiveX Control (excluding external COM objects)	8
Total Person Days	67

6.1.2. Registration Server

All time estimates have been given using Visual Basic as the development platform. It is possible (and likely) that some components will be developed on other platforms. It should be noted that these estimates are very general and can vary greatly during implementation.

6.1.2.1. Listener

Dependencies:

Database schemas for System Security database and Registration database; database connectivity to System Security database and Registration database (most likely ODBC if the databases are defined by NetRights); Object Registration and Object Service Manager components (in order to pass thread after user and object validation); DCOM (or Remote OLE automation or Microsoft TCP control)

User Interface:

Not required, but may be desired on server side in order to display open connections. Since there is a possibility connections will be dropped, the server side may need to manually close sockets.

Outstanding issues:

A protocol for communication with network. Until DCOM is available there are two possible solutions: Remote OLE Automation or TCP connection (by using the Microsoft TCP control); There is a possibility the Listener will have to interface with a legacy System Security database.

Estimated time needed for completion:

15-20 days

6.1.2.2. Object Services Manager

Dependencies:

Licensing Service component

User Interface:

None

Outstanding issues:

None

Estimated time needed for completion:

5 days

6.1.2.3. Licensing Service

Requirements:

Database schemas for Contracts database and Billing database; database connectivity to Contracts database and Billing database (most likely ODBC if the databases are defined by NetRights)

Dependencies:

Contracts and Billing Databases

User Interface:

None. UI for displaying contract terms, prices and accept/reject decision provided by desktop

Outstanding issues:

There is a possibility the databases used are legacy databases.

Estimated time needed for completion:

5-10 days.

6.1.2.4. Property Service

Requirements:

Schema for Registration database; ODBC for database connectivity

Dependencies:

Registration database

User Interface:

None

Outstanding issues:

None

Estimated time needed for completion:

5 days

6.1.2.5. Mail Service

Requirements:

Mail service on a machine accessible by the registration server

Dependencies:

Microsoft SMTP control (for sending mail)

User Interface:

None. UI provided by desktop.

Outstanding issues:

None

Estimated time needed for completion:

3 days

6.1.2.6. Object Registration

Requirements:

Schema for Registration database; ODBC for database connectivity; definition of a Template; structure of Template database

Dependencies:

Registration database, Template database

User Interface:

None.

Outstanding issues:

None.

Estimated time needed for completion:

5 days

6.1.2.7. Repository Handler

Requirements:

Structure of Object database; Network connectivity for interaction with web pages or other applications (Microsoft TCP control)

Dependencies:

Object database, JIT Packager

User Interface:

None

Outstanding issues:

None

Estimated time needed for completion:

5-10 days

6.1.2.8. Bulk Packager

Requirements:

Structure of Bulk database; Structure of Object database; GUI requirements (possibly)

Dependencies:

Bulk database, Object database, Packager

User Interface:

Possibly. This depends largely if a GUI is needed to perform bulk packing operations.

Outstanding issues:

Adding a GUI will increase development time

Estimated time needed for completion:

5 days without GUI, 10 days with GUI

6.1.2.9. JIT Packager

Requirements:

Schema for Registration database; ODBC for database connectivity; definition of a Template; structure of Template database

Dependencies:

Registration database, Template database, Packager

User Interface:

None.

Outstanding issues:

None

Estimated time needed for completion:

5 days

6.1.2.10. Packager

Requirements:

Definition of a Template; structure of Template database

Dependencies:

Template database

User Interface:

None

Outstanding issues:

None

Estimated time needed for completion:

5-10 days

6.1.2.11. Data Service

Requirements:

Structure of Referred Store database; Network connectivity for interaction with LIT Embedded control (Microsoft TCP control)

Dependencies:

Referred Store database

User Interface:

None

Outstanding issues:

None

Estimated time needed for completion:

3 days

6.1.2.12. Code Download Handler

Requirements:

Structure of Code Store database; Network connectivity for interaction with web pages or other applications (Microsoft TCP control)

Dependencies:

Code Store database, Internet Code Download API

User Interface:

None

Outstanding issues:

None

Estimated time needed for completion:

5 days

6.1.2.13. Registration database schema

Requirements:

Relational database (most likely Microsoft SQL Server); ODBC compatibility

Outstanding issues:

Definition of queries and transactions

Estimated time needed for completion:

10-15 days

6.1.2.14. System Security, Contracts, Billing database schemas

Requirements:

LicensIt-specific or legacy database. If the database is LicensIt-specific, it will most likely be a relational database (most likely Microsoft SQL Server) and ODBC compatible

Outstanding issues:

Defining an interface for Listener if a legacy database is used.

Estimated time needed for completion:

1 day each if using a LicensIt-specific database, 5+ days each if using a legacy database

6.1.2.15. Object, Template, Referred store, Code Store, Bulk database schemas

Requirements:

LicensIt-specific or legacy database. If the database is LicensIt-specific, it will most likely be some sort of storage (e.g. OLE structure storage, file system storage)

Outstanding issues:

Defining an interface for Listener if a legacy database is used.

Estimated time needed for completion:

1 day if using a LicensIt-specific database, 5+ days if using a legacy database

6.1.3. Tool Box

Template Creation:	Low	High
Bootstrap App hard coded to create a LIT Template	8	16
Creation of Hard coded "routines" called by controls (Update data, email, etc.)	4	8
Mapping Controls to Properties (Property Descriptors – giving names to properties)	3	6
Creation/specification of permissions	4	8
(Specification of Backstop Server?)	1	1
Specification/Binding of Connection object(s)	3	6
Creation of base input data file to be used by Express Packager	3	6
LPAPI (Registration and Packaging API)		
Read/instantiate Template for packaging apps	4	8
Instantiation of LIT Object	7	14
Persistence of LIT Object (read/write)	2	4
Registration of LIT Object	2	4
Validation routines (used by Tool Box and LIT Object):		
Validate UUID/Reg. number via checksum (MD5?)	3	6
Validate at Registry if network connected	1	2
Creation of UUID/Tool Box registration number		
Creation and/or validation of Tool Box registration number or LOB Unique ID	3	6
Creation/integration of Digital Signature	3	6
Express Packager		
Automate creation of LIT objects/LIT files		
Minimal UI to allow selection of template, input data file, content	4	8
Command line selection of template, input data file, content	2	4
Audit of all operations to log file(s)	3	6
Reports on content of log files	6	12
Communicate with LPAPI to instantiate selected template	1	2
Parse input data file into fields	3	6
Communicate with LPAPI to instantiate/register LIT objects	1	2
TOTAL:	71	141